Engineer

{ Desarrollo de backend y API }

# El motor de su aplicación o sitio web

La parte que el usuario no ve: dónde se guardan los datos, qué reglas se aplican y cómo pueden comunicarse otros sistemas con ella.

## ¿Aún está definiendo lo que necesita?

Cuéntenos qué está fallando o qué intenta construir. Le diremos qué haría falta, cuánto costaría aproximadamente y si somos las personas adecuadas.

**Desde 1.400 DKK por hora, sin IVA; y un precio por hora menor en una contratación más larga.**

**Póngase en contacto** Vea cuánto cuesta

Trabajos seleccionados que demuestran este servicio.

### Convierta el servicio en un resultado

Haga que «Desarrollo de backend y API» contribuya a un resultado empresarial. Podemos conectar el trabajo técnico con los usuarios, las operaciones y las decisiones a los que debe dar soporte. La parte que el usuario no ve: dónde se guardan los datos, qué reglas se aplican y cómo pueden comunicarse otros sistemas con ella.

**Póngase en contacto**

### Examine los trabajos relevantes

Utilice los trabajos relacionados de esta página para examinar nuestro enfoque. Pregunte por los compromisos entre objetivos y las condiciones detrás de cada resultado. Diseño y operación de 6 flujos ETL/ELT — 6 flujos ETL en tres motores de bases de datos

**Diseño y operación de 6 flujos ETL/ELT** Póngase en contacto →

### Diseño y operación de 6 flujos ETL/ELT

Diseñé, implementé y administré 6 flujos ETL/ELT con Google BigQuery, MSSQL, PostgreSQL, scripts de Shell, PL/pgSQL y Transact-SQL, integrando los datos para su procesamiento eficiente mediante una API en Python.

### 8 proyectos Power BI con manuales

Entregué 8 proyectos de Power BI con manuales completos, integrando las herramientas de Microsoft Power BI con una API en NodeJS y Python FastAPI para el análisis y la visualización de datos.

### Dirección del desarrollo integral de mapas GIS

Dirigí el desarrollo integral de mapas GIS, coordinando PostgreSQL, Mapbox, ReactJS y NodeJS para entregar una solución integrada.

### Más de 30 proyectos GIS: desarrollo y soporte

Dirigí el desarrollo, despliegue y soporte de más de 30 proyectos GIS con PostgreSQL, Bash, Python, JavaScript, GDAL, ArcGIS, PostGIS y Mapbox.

### Desarrollo de informes de análisis de datos

Desarrollé un sistema de informes de análisis de datos e incrementé en un 400 % los ingresos trimestrales del software mediante informes PDF generados con Python.

### Una plataforma marítima con límites claros entre sistemas

Diseñé una plataforma marítima por capas que separaba una interfaz PWA en Next.js, una API en Go (Huma/Fiber) y una capa de funciones PostgreSQL, manteniendo toda la lógica de negocio en la base de datos.

### Paso directo de JSON de Postgres a la API

Diseñé una arquitectura de paso directo de JSON en la que las funciones PostgreSQL devuelven documentos JSON completos que la API en Go transmite sin cambios, eliminando la deserialización intermedia y desacoplando la interfaz de los cambios del esquema.

### Cambio de contexto entre organizaciones

Diseñé un sistema de cambio de contexto entre organizaciones con localStorage en el cliente y una copia en cookies del servidor, que permite actuar en nombre de las organizaciones administradas y aplica autorizaciones de mínimo privilegio.

### API migrada a Huma v2 con paridad del 100 %

Migré la API HTTP de Fiber a Huma v2 — 649 rutas y 760 operaciones — y alcancé y mantuve una correspondencia del 100 % entre las rutas que registra el servidor y la descripción OpenAPI que publica.

### Contrato de API generado desde la base de datos

Generé el contrato de la API desde la base de datos hacia las capas externas — OpenAPI, un cliente TypeScript tipado de 44.076 líneas, 61 manejadores simulados y los límites que aplica la interfaz — con una comprobación en cada paso que falla ante cualquier divergencia.

### Correo con conmutación y registro de entregas

Desarrollé el correo electrónico como capacidad de la plataforma — tres proveedores con conmutación por error, webhooks de entrega, registros de envío y entrega, plantillas y campañas — con una comprobación de arranque que impide iniciar el sistema sin un proveedor.

### Datos PostgreSQL centrados en funciones

Diseñé una capa de datos PostgreSQL centrada en funciones — 1.275 funciones almacenadas en 34 esquemas — para que cada lectura y escritura pase por una función a la que la base de datos pueda conceder acceso, en lugar de hacerlo por una tabla.

### Combinación recursiva guiada por un catálogo

Implementé una combinación recursiva de preferencias JSON almacenadas, guiada por un catálogo, para evitar fallos por claves ausentes a medida que evoluciona el esquema.

### Aplicación reforzada con CSP basada en nonces

Reforcé la aplicación con CSP basada en nonces, HSTS, cookies SameSite, roles de base de datos de mínimo privilegio y nuevas comprobaciones de derechos de acceso en el servidor.

### Decisiones técnicas fundacionales de plataforma

Definí las decisiones fundacionales de la plataforma durante las semanas siguientes a la apertura del repositorio en noviembre de 2025 — la separación en capas, el acceso a datos centrado en la base de datos y el criterio de cero advertencias —, que seguían vigentes nueve meses después.

### Aplicaciones Django para IPTV

Desarrollé y mantuve aplicaciones web Django para la plataforma IPTV, entregando nuevas funcionalidades y mejorando el rendimiento y la estabilidad.

### Pruebas para Go, SQL y navegador

Desarrollé una batería de pruebas automatizadas por capas — 981 pruebas Go, 543 especificaciones de interfaz y navegador y 494 pruebas de comportamiento SQL — con pruebas de mutación, pruebas basadas en propiedades y un control obligatorio de accesibilidad.

### Stripe y acceso por compras en aplicaciones

Desarrollé la capa de pagos y derechos de acceso — Stripe junto con las compras dentro de las aplicaciones de Apple y Google —, regulando el acceso al directorio, la búsqueda y la exportación mediante un modelo de 11 tablas comprobado en el servidor.

### Trabajo lento trasladado a una cola River

Saqué las tareas lentas del procesamiento de solicitudes y las trasladé a una cola River — 15 módulos de trabajadores, 8 tareas programadas y 20 tareas pg_cron — para responder a las solicitudes mientras el trabajo continúa en segundo plano.

### Monitorización de errores y trazas propias

Desarrollé internamente la monitorización de errores y las trazas OpenTelemetry, en lugar de adquirirlas — saneamiento de cargas, detección de picos y regresiones, resolución de símbolos y una señal de actividad sintética —, con 11 vistas para operaciones.

### Límites de tasa y protección contra abusos

Desarrollé controles contra abusos que deniegan el acceso ante fallos — 22 limitadores de tasa con Redis, Cloudflare Turnstile, idempotencia de solicitudes y bloqueo del origen — para rechazar bots y avalanchas de solicitudes sin confiar en quienes las envían.

### Monedero de fidelidad, ligas e insignias

Desarrollé el sistema de fidelidad y reputación — 67 funciones sobre un registro de 31 tablas, con ligas, insignias y una tienda de canje —, bloqueando la fila del saldo para cerrar la ventana de doble gasto.

### Capa social: publicaciones, noticias y grupos

Desarrollé la capa social de la plataforma — publicaciones, noticias, grupos, menciones, un grafo de seguidores y un resumen de ocasiones — sobre la misma capa de datos centrada en funciones que el resto del producto.

## Espacios de contratación y carrera marítima

Construí espacios de contratación y de carrera profesional para marinos; la instantánea de implementación registrada contiene 151 funciones almacenadas en 48 tablas para vacantes, candidaturas, certificados, progresión de rango y tiempo de embarque.

## Generador de CV basado en base de datos

Desarrollé en Python un generador de CV, referencias, portafolios y cartas de presentación basado en base de datos — 41 módulos y 10.580 líneas — que genera seis formatos de salida desde una fuente SQLite de 23 tablas ensamblada mediante un flujo idempotente de 19 pasos.

## Generador: 981 pruebas y 92 % de cobertura

Mantuve el generador bajo 981 casos de prueba con un mínimo del 92 % de cobertura de ramas y las advertencias tratadas como errores, y verifiqué la idempotencia ejecutando dos veces todo el flujo de construcción desde un archivo vacío y exigiendo que la segunda pasada no cambiara nada.

## Cada regla del analizador tratada como error

Activé como errores todas las reglas del analizador Python y resolví 1.815 hallazgos hasta llegar a cero, con un motivo escrito para cada una de las escasas excepciones y dos de ellas respaldadas por una comprobación, además de un comentario.

## Codificador QR propio en 522 líneas

Incorporé al repositorio un codificador QR — Reed-Solomon sobre GF(256), disposición fija de módulos y ocho patrones de máscara — en 522 líneas para evitar una tercera dependencia de ejecución, y lo verifiqué leyendo la matriz final con un decodificador desarrollado de forma independiente.

## Todas las cadenas en un árbol de contenido

Trasladé todas las cadenas visibles para el usuario de Python a un árbol de 874 archivos de contenido en tres idiomas tras encontrar traducciones muertas cuya inutilidad no se veía y una comprobación que evaluaba silenciosamente solo un tercio de los logros.

## Concurrencia estricta Swift 6 sobre C

Adopté la concurrencia estricta completa de Swift 6 sin actores, conectando un bucle de eventos C bloqueante con el actor principal mediante un productor, un consumidor y un único orden, tras constatar que una tarea por evento pierde el orden del que depende la interfaz.

## Analizador que verifica cada llamada FFI

Escribí un analizador que lee la cabecera C real de 7.308 líneas y verifica cada punto de llamada, cada constante de enumeración y que toda clase propietaria de un puntero sea final, después de que una cabecera provisional escrita a mano permitiera compilar y enlazar llamadas a tres funciones eliminadas que causaban fallos al ejecutarse.

## La mitad sin utilizar del núcleo, implementada

Implementé la mitad del núcleo de mensajería que la aplicación no utilizaba — transferencia de copias de seguridad, mensajes temporales, edición y reenvío de mensajes, invitaciones verificadas, proxies y política de cifrado —, ejecutando cada prueba contra la biblioteca real, sin simulaciones.

## ¿Busca Desarrollo de backend y API?

Consulte los trabajos enlazados para valorar la experiencia relevante. Cuéntenos qué necesita construir; podemos hablar del alcance, el enfoque de entrega y el traspaso.

**Póngase en contacto** Portafolio completo

### Defina el alcance antes de encargar el trabajo

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

**Vea cuánto cuesta** Póngase en contacto →

# Conversemos sobre sus requisitos

Cuéntenos qué necesita de «Desarrollo de backend y API», incluidos el plazo y el equipo que lo mantendrá. Podemos hablar de una forma práctica de avanzar.

**Póngase en contacto**

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

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/services/backend-development/