Engineer Software

Automatización y CI/CD

35 logros

Conecte las partes de su sistema

Los trabajos reunidos en «Automatización y CI/CD» muestran una parte de un sistema en funcionamiento. Podemos ayudar a conectar esa parte con el resto de su operación. Las cifras llegan donde hacen falta según un calendario, sin que nadie tenga que copiar una hoja de cálculo al terminar el mes.

Explore nuestros servicios Póngase en contacto →

Encuentre un reto comparable

Siga un ejemplo que se parezca a su situación. Su reseña describe las decisiones detrás del resultado, más allá de la etiqueta tecnológica. Un plan de infraestructura para 14 departamentos — 14 departamentos de investigación con un único plan de infraestructura

Un plan de infraestructura para 14 departamentos Póngase en contacto →

1.000 incidencias de datos geográficos resueltas

Resolví 1.000 incidencias en datos geográficos y series temporales con GDAL, ArcGIS, PostGIS, Mapbox, QGIS, SQL (PL/pgSQL, Transact-SQL) y Bash, para mantener la calidad del procesamiento de grandes volúmenes de datos.

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.

Despliegue GIS SaaS con GitHub Actions

Automaticé el despliegue de una aplicación GIS SaaS, el procesamiento de datos y el sistema de informes con GitHub Actions CI/CD, Python, Bash y SQL.

Entrega automática de 20 flujos de datos GIS

Automaticé la entrega de 20 flujos de datos GIS y procesos ETL de datos de aplicaciones, simplificando la automatización de la infraestructura y la elaboración de informes.

Procesos renovados: 8.000 horas ahorradas

Reorganicé los procesos internos y ahorré 8.000 horas mediante mejoras en la arquitectura del software, los sistemas y la planificación.

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.

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.

i18n en inglés y danés

Establecí un sistema de internacionalización en inglés y danés con 12.027 mensajes por configuración regional distribuidos en 458 archivos de espacios de nombres, con vocabulario controlado por análisis estático y un límite de 400 líneas por archivo.

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.

Motor de 268 controles para el repositorio

Desarrollé el motor de comprobaciones del repositorio — 268 controles de commit registrados, 277 reglas de análisis estático y 15 reglas ESLint propias —, además de 146 pruebas de los propios controles, para que la compilación haga cumplir el criterio de calidad.

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.

Su código en su propio servidor

Desplegué la forja git propia de la empresa con Soft Serve, privada por defecto, sin panel web y con el puerto SSH limitado a loopback detrás de un servidor de salto, e hice de su página de entrada un artefacto de compilación del sitio principal en lugar de una copia mantenida a mano.

Control de commits con 22 analizadores

Desarrollé un control obligatorio de commits con 22 analizadores de una línea y cinco que requieren una explicación, sin nivel de advertencias ni supresiones en línea permitidas, que cubre HTML, CSS, JavaScript, Python, YAML, Markdown, Shell, enlaces, ortografía, secretos y tipografía.

Una red de seguridad que ya no cuesta media hora

Reduje el tiempo del control de calidad del sitio basado en navegador de 1.636 segundos a 615 al ordenar las comprobaciones de mayor a menor duración y ejecutarlas en un grupo limitado a cuatro trabajadores, tras medir que el orden alfabético requería 320 segundos frente a 224.

27 tamaños de fuente por una escala de 6 pasos

Sustituí 27 tamaños tipográficos renderizados cuyos vecinos más próximos diferían un 0,6 % por una escala de tercera mayor de seis pasos, y escribí el analizador que rechaza un vigesimoctavo tamaño.

Los fallos de accesibilidad que el verificador no detecta

Detecté y corregí 11 defectos de accesibilidad que el navegador presentaba como correctos: ocho enlaces de pie de página que seguían en el orden de tabulación pese a pointer-events, una línea temporal de desplazamiento que recortaba el colofón en tres páginas y un motor que ejecutaba 70 de sus 105 reglas.

Sistema de color cerrado en 28 colores

Delimité el sistema de color a 28 colores documentados con un analizador que rechaza valores usados pero no documentados, documentados pero no usados, mal medidos, reescritos como literales o situados a una distancia perceptiva de 0,02 de otro existente.

Una capa completa ausente del tema claro

Detecté que al tema claro le faltaba una capa de superposición de página completa desde su creación al comprobar que ambos temas pintaran cada superficie compuesta con el mismo número de capas.

Espejos del sitio en Gemini y Gopher

Construí salidas Gemini y Gopher a partir de la misma fuente del sitio web; el logro registra 777 documentos en cada formato en aquella etapa, con cero bytes de HTML modificados al añadir los formatos.

963 bloques de metadatos por un único grafo

Sustituí 963 bloques anónimos de datos estructurados que repetían la empresa 1.671 veces por un único grafo enlazado de 16 tipos con identificadores de origen estables.

Paquete CSS reducido de 87 KB a 31 KB

Reduje el paquete de estilos de 87 KB a 31 KB al retirar una fuente base64 y eliminé 756 KB en 22 archivos que se publicaban en cada despliegue sin que nada los referenciara.

Sitemap corregido: fechas que eran iguales

Corregí un sitemap en el que 172 de 176 URL compartían una misma fecha de modificación, tomando la fecha del historial git tras constatar que la exportación reescribe todos los archivos en cada ejecución.

Control de calidad de 10.242 líneas analizado

Apliqué un formateador y un analizador a 10.242 líneas de JavaScript del control de calidad tras constatar que eran el mayor bloque de código del repositorio y el único que ninguna herramienta revisaba; corregí 13 hallazgos sin suprimir ninguno.

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.

106 de 106 comprobaciones del perfil PDF/UA-1

Registré la validación PDF/UA-1 en nueve documentos, con 106 de 106 comprobaciones superadas en el perfil de accesibilidad configurado, mientras la variante de archivo incumplía una de 146 reglas.

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.

Pruebas para los propios verificadores

Escribí pruebas para los propios verificadores tras constatar que un verificador que solo recibe entradas correctas puede acabar informando de éxito por no haber leído nada; introduje una falta de ortografía para confirmar su detección y obtuve un intervalo de identificadores de la base de datos en vez de fijarlo en la prueba.

Comprobación de cifras entre idiomas

Desarrollé una comprobación de contenido entre idiomas que falla si una traducción omite una cifra del inglés o usa una notación ajena al idioma; detectó dos descripciones danesas sin una métrica y dieciséis pasajes ucranianos con comillas del estilo inglés.

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.

17 iconos con nombre para lectores de pantalla

Asigné nombres accesibles a todos los controles de la interfaz compuestos solo por un icono tras encontrar el botón de envío anunciado como «arrow up circle, button», y escribí el analizador que exige una etiqueta a no más de ocho líneas del icono.

Sistema de diseño macOS de 48 tokens

Desarrollé un sistema visual de 48 tokens sobre el material de vidrio propio de la plataforma y escribí el analizador que rechaza números sin significado declarado, tamaños de fuente fijados directamente y animaciones que ignoran la preferencia de movimiento reducido.

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.

Identidad visual creada a partir de 2 dibujos

Desarrollé los conjuntos de iconos y favicons de la empresa a partir de dos dibujos originales, con todos los recursos derivados regenerados mediante scripts y una comprobación que falla si se confirma un archivo derivado anterior a su fuente.

Seis recolectores declaran lo que omitieron

Hice que seis recolectores de informes declararan la parte de sus entradas que nunca habían leído — archivos no abiertos, sondeos no instalados y líneas del diario no reconocidas — y ajusté los umbrales a 116 muestras medidas tras constatar que una recolección ciega y un servidor sano generan la misma página.

Centre el esfuerzo de ingeniería

Traiga un conjunto de datos representativo, las preguntas que debe responder y el cuello de botella actual. Podemos acordar cómo medir la calidad, la capacidad de procesamiento y la recuperación.

Vea cuánto cuesta Póngase en contacto →

Sitúe el tema en su contexto

Cuéntenos cómo aparece «Automatización y CI/CD» en su propio sistema. Podemos ayudarle a elegir el servicio relevante y definir el trabajo en torno a él.

Explore nuestros servicios

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://software.engineer.company/es/categories/automation/