Un botón para no depender del técnico

Una tienda online sincroniza con FactuSOL cada pocos minutos, pero ver un cambio al momento exigía pedírselo a la única persona que podía migrar los datos.

Esta es la historia de cómo ese «¿me lanzas la migración?» se convirtió en un botón. Un botón sencillo para quien lo pulsa, que detrás esconde un problema de arquitectura, cuatro revisiones independientes y varias sorpresas que solo aparecieron en producción.

El cuello de botella

La sincronización entre FactuSOL y la tienda la hace el conector de Festeweb. Tiene migraciones de varios tipos, programadas a intervalos regulares: ni tan frecuentes que saturen el servidor, ni

tan espaciadas que compliquen el trabajo del personal de tienda. También tiene un botón para lanzarlas a mano, pero está dentro del propio programa, que solo puede estar instalado en un ordenador, y a ese ordenador solo accede el técnico.

El equilibrio funcionaba, pero faltaba algo: un botón para no depender del técnico cuando surge una necesidad puntual. Por ejemplo, antes de preparar una campaña publicitaria, cuando los cambios del catálogo tienen que estar en la web ya, no en la siguiente migración programada. Un botón al alcance del personal, desde cualquier navegador, sin entrar en ese ordenador.

Un conector, un solo PC

La solución obvia era instalar el conector en más ordenadores. Ya se había probado en varios y no funcionó: cada instalación guarda su configuración en su propio equipo, así que los cambios hechos en una no llegaban a la otra, y las comprobaciones que hacía cada una al migrar entraban en conflicto con las de la otra. El conector tiene que estar en un solo sitio.

Por eso el conector vive en un único miniPC encendido día y noche, al que el técnico accede por escritorio remoto.

Dar ese mismo acceso al personal de tienda habría sido sencillo de configurar, pero un riesgo serio. Un escritorio remoto no entrega un botón, entrega el ordenador entero: el conector con sus credenciales de FactuSOL y de la tienda, y todo lo demás que hay en ese equipo, incluidos datos protegidos de este y de otros clientes. Un clic equivocado, una contraseña compartida o un equipo de la tienda comprometido habrían bastado para exponerlo todo. Además, cada persona con acceso multiplica las puertas de entrada que hay que vigilar.

Un escritorio remoto para pulsar un botón es un lanzallamas para encender una vela.

La restricción quedó clara: el conector no se puede duplicar, el PC está detrás del router de una oficina y no conviene abrir puertos. Y la tienda necesita lanzar la migración desde cualquier navegador.

La web no llama, el PC pregunta

La clave fue invertir la dirección de la comunicación. Si la web no puede llegar al PC, es el PC quien pregunta a la web.

El conector de Festeweb permite añadir un módulo de personalización en Python que carga al empezar a trabajar. Se estudió cómo carga esos módulos y se aprovechó ese punto para añadir un pequeño hilo que, cada 30 segundos, envía un «latido» a una página web privada: estoy vivo, estoy libre u ocupado. Si en ese momento hay una petición pendiente, la recoge, lanza la misma migración que el botón del conector y avisa de cuándo empieza y cuándo termina.

Pulsar el botón no ejecuta nada: solo deja una petición en cola. Es el conector quien decide si puede atenderla.

En la web, una página con acceso por nombre y contraseña muestra el estado y el botón. No hay que abrir ningún puerto, no se toca el software del proveedor y el conector sigue siendo el único que habla con FactuSOL. Además, como usa el mismo mecanismo interno que las tareas programadas, una migración manual nunca se ejecuta a la vez que una automática.

Frenos para las prisas

Dar un botón a quien quiere ver los cambios ya tiene un riesgo evidente: pulsarlo una y otra vez. Y la migración del catálogo pesa sobre el servidor de la tienda.

Por eso el botón solo se activa si se cumplen varias condiciones: el conector responde, no hay otra sincronización en curso y la carga del servidor está por debajo de un umbral. La carga se mide con el *load average* de un minuto, comparado con el número de núcleos del servidor:

- **Tranquila:** por debajo de tres cuartas partes de los núcleos.
- **Con carga:** hasta vez y media los núcleos. Se puede migrar, pero la página recomienda esperar.
- **Mucha carga:** por encima. El botón se bloquea.

No importa el motivo: una campaña de marketing, un mantenimiento o un rastreo masivo de buscadores o de inteligencias artificiales. Si el servidor sufre, la migración manual espera. Tras una migración correcta, además, hay una pausa de dos minutos antes de permitir otra.

Una precisión importante: este freno solo afecta a las migraciones manuales. Las programadas del conector siguen su calendario.

Lo que vieron las auditorías

Con la primera versión funcionando, el código pasó por revisiones independientes de seguridad, usabilidad, compatibilidad entre navegadores y comportamiento ante fallos. En total se revisaron más de 40 puntos. Algunos hallazgos sirven para cualquier proyecto:

- **El registro también es una superficie de ataque.** El nombre de usuario de un intento de acceso fallido se escribía tal cual en el log. Con un salto de línea en ese campo, un atacante podía fabricar líneas falsas, como un acceso correcto que nunca ocurrió. Y sin límite de tamaño, podía llenar el disco a base de envíos.
- **El tiempo también delata.** La comprobación de contraseñas tardaba distinto según el usuario existiera o no, lo que permitía adivinar nombres válidos.
- **Un mensaje perdido es un estado incoherente.** Si el aviso de final no llegaba a la web por un corte de red, la página daba la migración por perdida aunque hubiera terminado bien. Ahora el conector repite ese aviso con cada latido hasta que la web confirma que lo ha recibido.
- **Las plantillas también son un riesgo.** Si la clave de ejemplo de la configuración no se cambiaba, la web la aceptaba. Ahora la rechaza siempre.

Ninguno de estos fallos impedía que el botón funcionara. Por eso no se habrían detectado sin buscarlos.

Lo que solo aparece en producción

Las pruebas locales no lo cubren todo. Al instalarlo en el conector real aparecieron errores que ninguna simulación habría mostrado, y otros que eran simplemente suposiciones equivocadas:

- **Errores que no son errores.** El conector registra con la palabra «ERROR» algunos elementos que no puede guardar, pero sigue adelante y termina bien. La primera versión los interpretaba como fallo de la migración. Ahora distingue entre un fallo real y un elemento rechazado, y solo cuenta estos últimos.

- **Python no relee lo que ya ha cargado.** Se dio por hecho que el conector recogería el módulo modificado en la siguiente tarea. No es así: el intérprete conserva en memoria la versión anterior hasta que se reinicia el programa.
- **Unos milisegundos cambian el orden.** El módulo comprobaba el latido antes de que el conector limpiara su ventana. Ese pequeño retraso bastaba para que la limpieza llegara a mitad de la migración y borrara lo ya escrito. Invertir el orden lo resolvió.
- **La configuración no debe vivir en el código.** La clave de comunicación iba dentro del módulo y se perdía cada vez que se pegaba una versión nueva. Ahora se lee del archivo de configuración del propio conector.
- **Cada entrega, un número de versión.** Durante las correcciones se entregaron archivos distintos con la misma versión. Cuando algo falla, saber exactamente qué está instalado es lo primero, y un número repetido lo impide.

Pensado para quien lo usa

Quien pulsa el botón no es técnico. Eso guió cada decisión de diseño: textos sin jerga, letra grande, colores con contraste suficiente y una regla de «menos es más». La carga del servidor no se muestra en cifras, sino como una barra y una palabra.

La mejor interfaz es la que no hay que explicar.

Los estados se cuentan de forma directa: disponible, en cola, migrando, completada o el motivo concreto por el que ahora no se puede. Mientras migra, una barra avanza según la duración media de las últimas migraciones, de modo que da una idea realista de cuánto falta. Y una línea punteada, que se rellena segundo a segundo hasta el siguiente latido, indica que el conector sigue vivo.

La página se comprobó en 16 tamaños de pantalla (móvil en vertical y en horizontal, tablet, portátil y sobremesa) y en 11 situaciones distintas: 176 combinaciones sin barras de desplazamiento y sin elementos que salten al cambiar de estado.

Lecciones aplicables

- **Si un sistema no puede recibir llamadas, puede hacerlas.** Un latido periódico resuelve muchos problemas de conectividad sin abrir puertos.
- **Un botón para usuarios no técnicos necesita frenos.** Comprobar el estado del sistema antes de actuar evita que la comodidad se convierta en un problema.
- **Lo importante se confirma.** Toda comunicación entre sistemas debe confirmar la recepción de los mensajes clave, o tarde o temprano habrá estados incoherentes.
- **La seguridad no está solo en la contraseña.** Los registros, los tiempos de respuesta y las plantillas de configuración también cuentan.
- **Producción manda.** Las pruebas locales son necesarias, pero el comportamiento real del software de terceros solo se conoce instalándolo.

Cierre

Lo que para el usuario es un solo botón suele esconder más trabajo que las funciones complejas. El resultado es una herramienta pequeña que elimina una dependencia diaria: el personal de tienda actualiza

la web cuando lo necesita, y el servidor queda protegido de las prisas.

Si te ha gustado este PDF, compártelo. Si no te ha gustado, díselo a Andy.