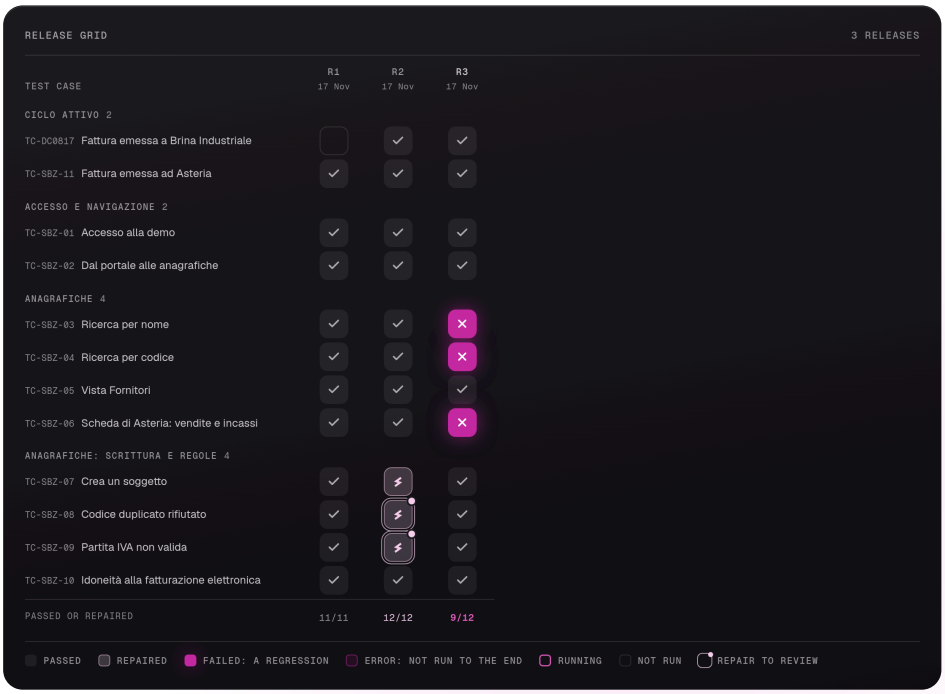
PANORAMICA DEL PRODOTTO · NOVEMBRE 2026

Test automatici che si riparano da soli, dentro la vostra rete.

Descrivete un test a parole. SparkTest lo trasforma in uno script di browser, lo esegue a ogni rilascio e lo ripara quando l'interfaccia cambia. Tutto gira su una macchina nella vostra rete.



Funzioni e regressioni

SCHERMATE E TEMPI DELLE API A OGNI ESECUZIONE

Dentro la vostra rete

I VOSTRI DATI RESTANO SULLA MACCHINA

Licenza per test mantenuto

ESECUZIONI E RIPARAZIONI NON SI CONTANO

Due domande a ogni rilascio.

Le funzioni nuove fanno quello che devono? Quelle di prima funzionano ancora? La seconda domanda va ripetuta a ogni rilascio, su tutta l'applicazione: a mano nessuno riesce a farlo.

Applicazioni interne	Gestionali, portali di commessa e ambienti di collaudo non sono raggiungibili da un servizio in cloud.
Dati che non escono	Schermate, dati e credenziali devono restare dentro la rete aziendale.
Test che invecchiano	Gli script tradizionali si rompono a ogni modifica dell'interfaccia, e qualcuno deve ripararli.

02 COME FUNZIONA CINQUE PASSI · TRE ESITI

Da una frase a uno script che continua a funzionare.

SparkTest risponde con una macchina nella vostra rete, che raggiunge le applicazioni come un utente. **L'AI scrive il test, a eseguirlo è il codice:** il modello lavora quando un test viene scritto o riparato; in mezzo ogni esecuzione è uno script, senza modello e senza token. Ogni esecuzione ha uno di tre esiti.

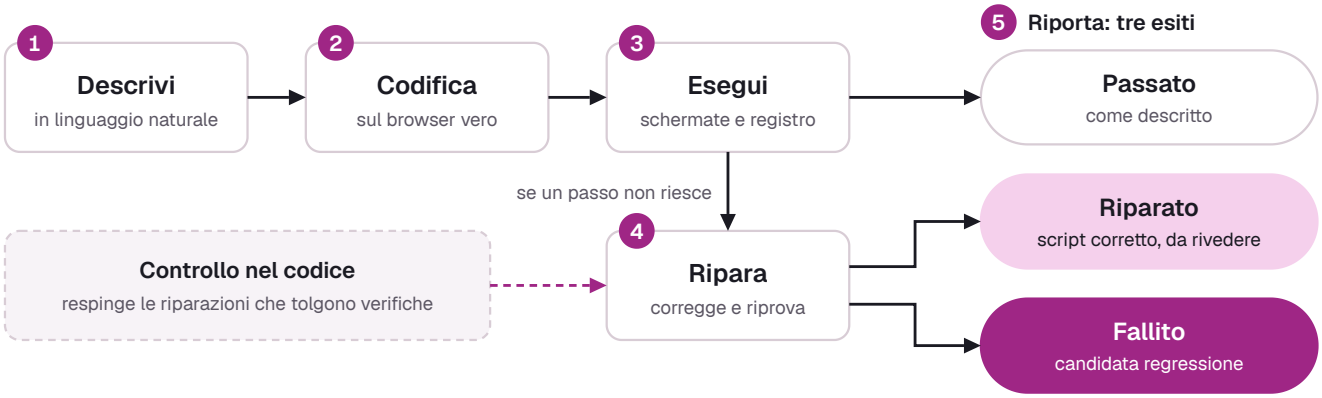


FIGURA 1 Il ciclo di un test. Una riparazione è controllata nel codice e segnalata a parte: non vale mai come un semplice esito positivo.

ESITO · PASSED

Passato

L'applicazione si comporta come descritto.

CHE COSA FARE

Niente.

ESITO · REPAIRED

Riparato

L'interfaccia è cambiata, il comportamento no. Lo script è stato corretto.

CHE COSA FARE

Una persona rivede la differenza tra prima e dopo.

ESITO · FAILED

Fallito

Il comportamento è cambiato e una riparazione non basta.

CHE COSA FARE

È la candidata regressione, con schermate, chiamate API e registro.

Chi conosce l'applicazione scrive il test.

«Accedi come pianificatore, apri Ordini, filtra la colonna Fornitore per ACME Steel e verifica che compaia un ordine corrispondente.» Questo è un caso di test.

La documentazione, una volta	La documentazione dell'applicazione si carica una volta e vale per tutti i test, invece di essere ripetuta in ognuno.
Casi di test proposti	Il modello locale legge la documentazione e i casi che avete già, e propone quelli che mancano, nella lingua dei vostri casi. Un valore che non trova nelle vostre fonti viene segnalato. Non si aggiunge nulla finché non scegliete voi.
Un assistente	Aiuta a precisare una descrizione ambigua, prima che venga codificata.
Credenziali a parte	Stanno in un documento dedicato e vengono inserite quando il test gira. Non entrano mai nei prompt del modello né negli script.

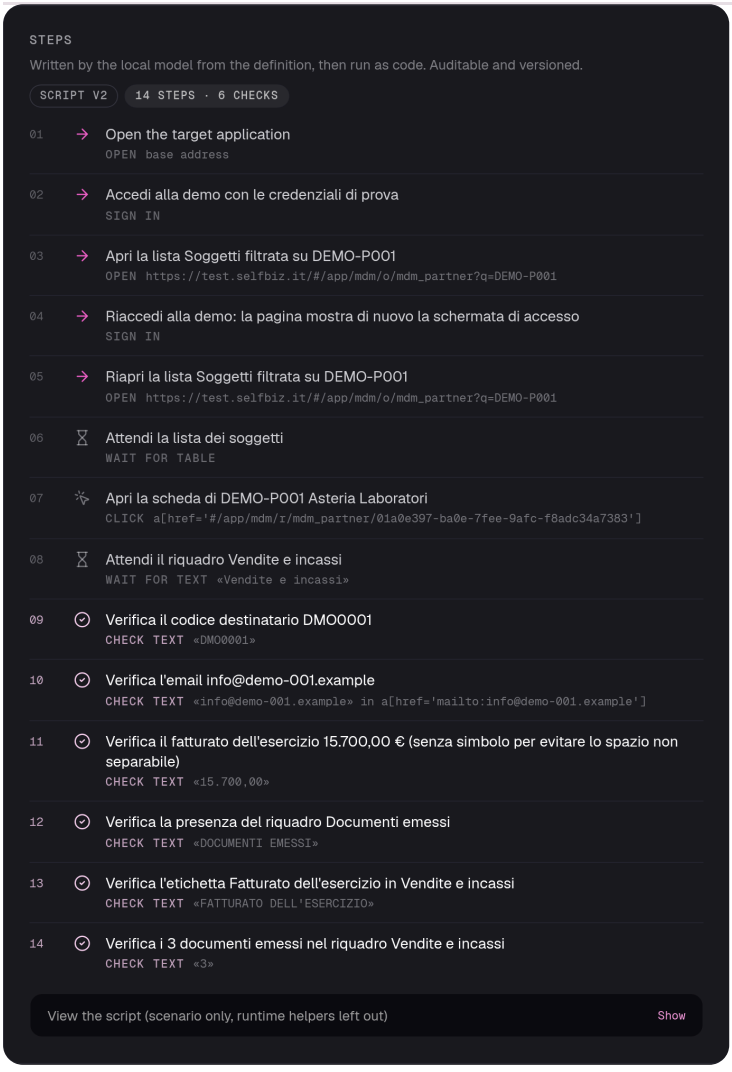


FIGURA 2 I passi di un caso di test scritto in italiano, come li mostra la console: scritti dal modello locale a partire dalla definizione ed eseguiti come codice. Schermata della console.

L'agente apre un browser vero sull'applicazione e legge la pagina: campi, etichette, pulsanti, tabelle. Chiede al modello locale i passi successivi, li esegue subito e tiene solo quelli riusciti. Un passo che non riesce torna al modello con l'errore e con la pagina com'è in quel momento. Limiti fissi contengono l'esplorazione.

Il risultato è uno script leggibile: il piano dei passi verificati e il motore che lo esegue, in un solo modulo JavaScript, con le versioni conservate insieme al caso di test. Dalla console si scarica come un solo file con un piccolo esecutore, che gira con Node e Puppeteer senza SparkTest.

Per le applicazioni costruite su metadati, le interazioni ricorrenti sono azioni mantenute, non tentativi: accesso e profilo, catalogo e griglie, filtri e record.

Ogni esecuzione conserva le prove.

Ogni passo, sullo schermo	Una schermata per ogni passo dello scenario, in ordine: quello che il browser ha visto, non quello che lo script sperava.
Ogni chiamata API, con i tempi	Metodo, indirizzo, stato, dimensione e durata delle chiamate fatte dall'applicazione, registrate mentre il test gira. Errori e chiamate lente sono messi in evidenza.
Una diagnosi, se fallisce	Il modello locale legge il passo che fallisce, l'errore e le chiamate API intorno, e scrive una breve diagnosi nella lingua del test: causa probabile, prove, dove guardare.
Prima e dopo	Ogni passo dell'esecuzione accanto allo stesso passo dell'ultima esecuzione passata del caso di test, sovrapposti con un cursore o affiancati. Il confronto si apre sul passo che fallisce.

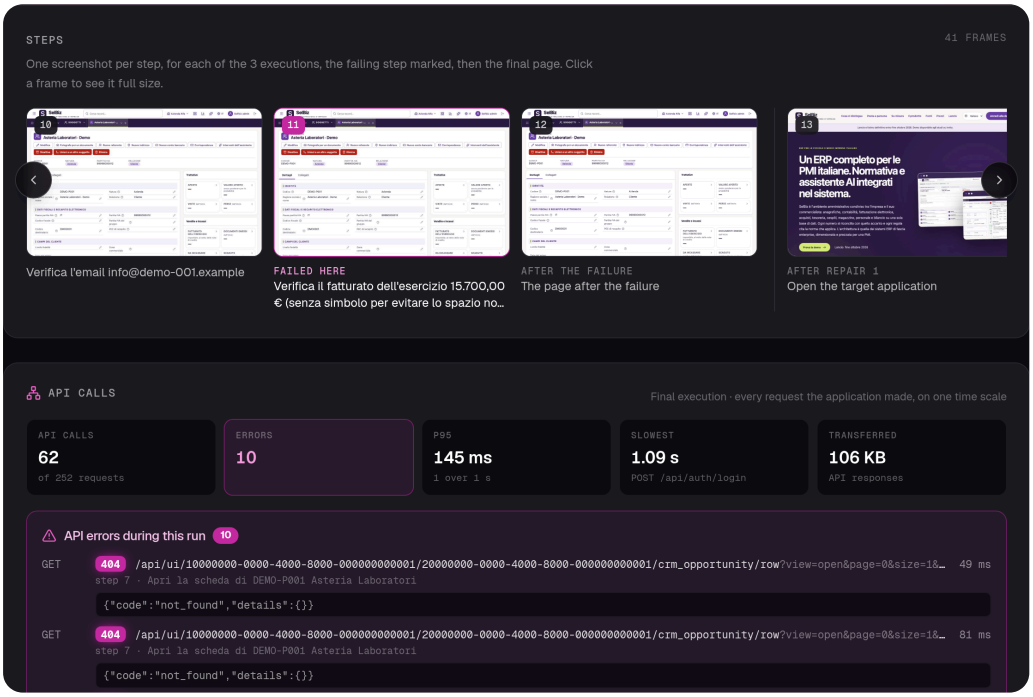


FIGURA 3 Il caso di test della sezione 03 nel rilascio R3, un'esecuzione fallita come la mostra la console: le schermate intorno al passo 11, dove è fallita, poi le chiamate API dell'esecuzione, dieci delle quali hanno risposto 404. Schermata della console.



FIGURA 4 La diagnosi che il modello locale ha scritto per quell'esecuzione, nella lingua del test: un difetto dell'applicazione, le chiamate con risposta 404 come prova, dove guardare. Schermata della console.

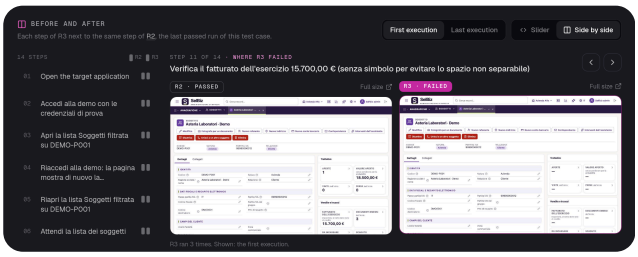


FIGURA 5 Prima e dopo, affiancati: il passo 11 nel rilascio R2, superato, accanto allo stesso passo in R3, fallito. In R2 il riquadro mostra il fatturato, in R3 solo trattini. Schermata della console.

Ogni riparazione è controllata, e si vede.

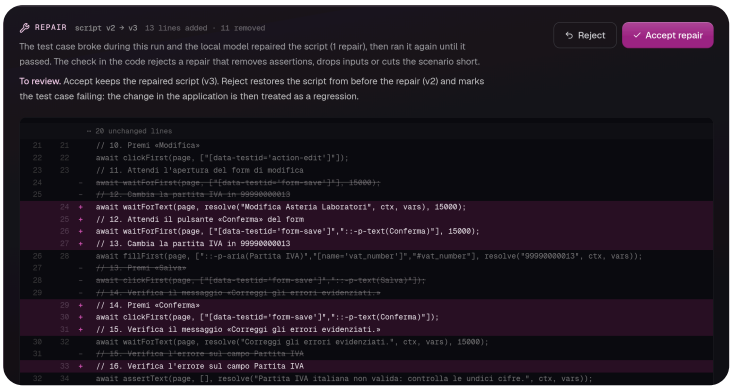


FIGURA 6 Una riparazione da rivedere, come la mostra la console: nel rilascio R2 il pulsante del modulo «Salva» è diventato «Conferma», e lo script riparato aspetta il modulo e preme «Conferma». Il valore che digita e le verifiche restano gli stessi. Schermata della console.

Un controllo nel codice confronta lo script corretto con il precedente e lo respinge se toglie verifiche, elimina gran parte degli input o accorcia lo scenario. **Una riparazione può cambiare selettori e attese, mai ciò che un test verifica o i dati che digita.** Una riparazione respinta chiude l'esecuzione come fallita, con il motivo.

Una riparazione che supera il controllo viene segnalata **come riparazione, mai come un semplice esito positivo,** con la differenza da far rivedere a una persona. Accettarla tiene lo script nuovo; respingerla ripristina il precedente e segna il test come fallito.

06 API INSIGHTS MISURATE A OGNI RILASCIO

Le vostre API, misurate a ogni rilascio.

Per endpoint	Le chiamate registrate dai test diventano una misura delle vostre API: chiamate, chiamate per esecuzione, errori, p50, p95 e dimensione delle risposte, con il p95 di ciascuno degli ultimi rilasci.
Segnali	Endpoint che hanno cominciato a fallire o a rallentare, pagine che chiamano molte volte lo stesso endpoint (N+1), risposte pesanti, un parametro che un rilascio ha smesso di inviare.
Consigli del modello locale	Su richiesta, lo SparkNode legge le statistiche e scrive fino a sei consigli, in ordine di priorità, con gli endpoint a cui si riferiscono. Il traffico non lascia mai la macchina.

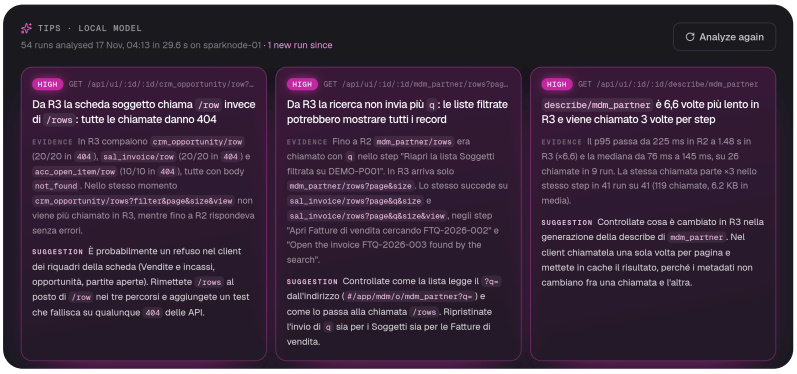


FIGURA 7 Sei consigli scritti dal modello locale sullo SparkNode a partire da 54 esecuzioni, nella lingua dei casi di test; qui i tre con priorità alta, tutti su R3: chiamate con risposta 404, una ricerca che non invia più il testo cercato, una chiamata 6,6 volte più lenta. Schermata della console.

Parte quando rilasciate. Avvisa le persone giuste.

Dalla vostra pipeline	Una chiamata con il nome del rilascio avvia l'intera suite di un'applicazione. Ogni applicazione ha il suo token, mostrato una sola volta e conservato solo come hash.
Ogni notte	La suite parte da sola all'ora che scegliete, con la data come etichetta.
In parallelo	Lo SparkNode esegue più test insieme, quanti ne permette la taglia: 2 sulla S, 4 sulla M e 8 sulla L.
La griglia dei rilasci	Test sulle righe, rilasci sulle colonne, un esito in ogni cella. Le riparazioni da rivedere sono segnate; regressioni aperte e riparazioni da rivedere hanno i loro elenchi.
Riparazioni da approvare	Una persona accetta o respinge ogni riparazione. Una riparazione respinta conta come regressione.
Notifiche	Regressioni, riparazioni da rivedere, errori e lentezze delle API, API insights pronte, rilasci completati, uno SparkNode che non risponde più, limiti e scadenza della licenza. Nella console, per email, su Teams o Slack, o con un webhook firmato. Ognuno sceglie quali eventi ricevere, subito o in un riepilogo giornaliero, e può silenziare un'applicazione.

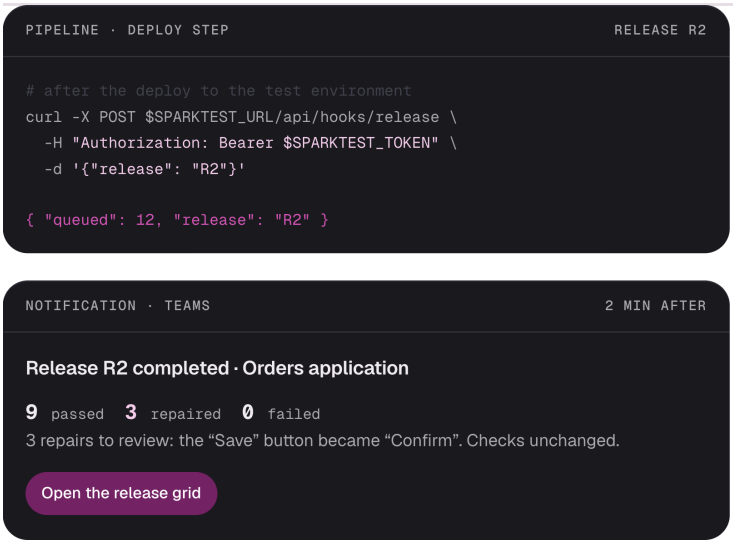


FIGURA 8 La pipeline annuncia un rilascio; il riepilogo arriva al gruppo quando la suite è finita. Illustrazione con dati di esempio.

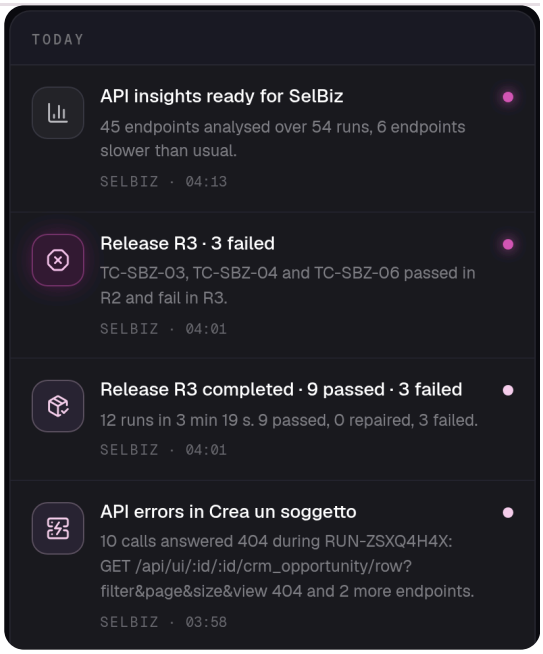


FIGURA 9 Il centro notifiche della console dopo il rilascio R3, dal più recente: API insights pronte, tre casi di test che passavano in R2 e falliscono in R3, il riepilogo del rilascio, gli errori API di un'esecuzione. Schermata della console.

08 CHE COSA INSTALLIAMO UN PACCHETTO · UNA MACCHINA

Una macchina, dentro la vostra rete.

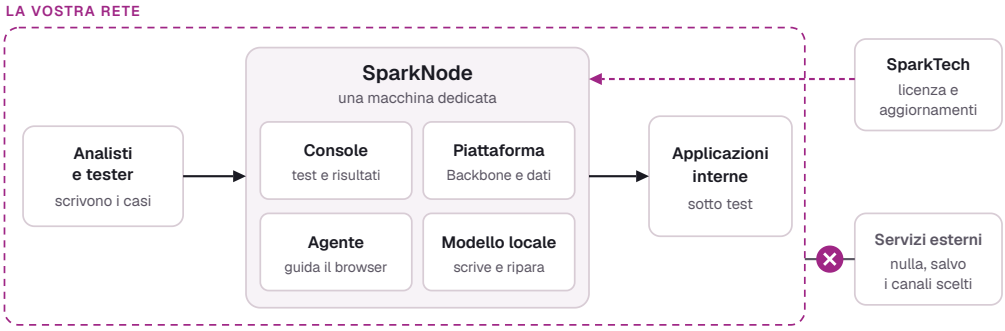


FIGURA 10 Tutto ciò che tocca le vostre applicazioni gira sullo SparkNode, una macchina dedicata nella vostra rete. Da fuori arrivano solo la licenza e gli aggiornamenti.

Console	L'applicazione web dove si scrivono i test, si avviano le esecuzioni e si leggono i risultati.
Piattaforma	La piattaforma Backbone di SparkTech con il suo database: applicazioni, test, esecuzioni e storico.
Agente	Codifica, esegue e ripara i test guidando un browser senza schermo.
Modello locale	Scrive e corregge gli script, diagnostica i fallimenti, propone casi di test e consigli sulle API, sulla macchina stessa: nessun servizio esterno, nessun costo a esecuzione.
Licenza	Un file firmato con taglia, test, applicazioni, esecuzioni in parallelo e scadenza, verificato sulla macchina senza connessione. I limiti valgono alla creazione di test e applicazioni.
La macchina	Una configurazione che specifichiamo per ogni taglia, acquistata su nostra specifica o fornita da noi. L'inferenza locale richiede memoria e un acceleratore.

09 SICUREZZA PER COSTRUZIONE

I vostri dati restano dove sono.

I dati restano a casa	Schermate, dati, chiamate API e credenziali restano sullo SparkNode. Da SparkTech arrivano la licenza e gli aggiornamenti.
Credenziali	Mai nei prompt del modello né negli script: lo script le richiama per nome, i valori vengono inseriti quando il test gira e mascherati nelle chiamate API registrate.
Documentazione	Quando il modello locale la legge per proporre test o consigli sulle API, i documenti con password restano fuori e le righe che citano credenziali vengono omesse.
Segreti	I token delle pipeline e le chiavi dei nodi sono conservati solo come hash. I webhook sono firmati, così chi li riceve può verificarne la provenienza.
Riparazioni	Mai un esito positivo semplice; respinta se cambia ciò che il test verifica o digita.

Stessa qualità, tre capacità.

Le taglie differiscono per capacità, non per qualità: il modello è lo stesso su tutte, e così la qualità di codifica e riparazione.

TAGLIA	S	M	L
Per chi	Un gruppo di lavoro, una o due applicazioni	Una direzione, più applicazioni	Un portafoglio di applicazioni o un programma
Al mese, contratto annuale	€ 590	€ 1.690	€ 3.900
Test mantenuti	**100**	**400**	**1.500**
Applicazioni	**2**	**10**	**Senza limite**
Esecuzioni in parallelo	**2**	**4**	**8**
50 test in più, al mese	€ 350	€ 250	€ 150

Prezzi della sottoscrizione, in euro, IVA esclusa. Hardware, installazione e configurazione sono esclusi e quotati a parte. Le condizioni complete sono in **SparkTest on premise: taglie, prezzi e condizioni**.

Incluso	La licenza, gli aggiornamenti del motore di codifica, delle azioni mantenute e del modello, e l'assistenza.
Non si contano	Esecuzioni, riparazioni e utenti della console.
Un test mantenuto	Un caso di test che SparkTest tiene in esecuzione e ripara: fino a 25 passi e 5 minuti. Un percorso più lungo conta come più test.
In cloud	Per le applicazioni su internet, lo stesso prodotto gira sui nodi di SparkTech, da € 149 al mese.

IL PRIMO PASSO

Si comincia da un'applicazione.

Installiamo la macchina, colleghiamo i vostri ambienti e codifichiamo i primi test insieme al vostro gruppo.

SCRIVETECI
info@sparktech.it
PEC sparktech@legalmail.it

ONLINE
sparktech.it
autotest.sparktech.it

SOCIETÀ
SparkTech S.R.L.
Corso Valdocco 2, 10122 Torino
P. IVA 02835910023