SPARKFABRIK

SPARKFABRIK **WHITE PAPERS**

GUIDA ALLA **DEVELOPER EXPERIENCE**

Come e perché fare della DX il tuo asset strategico

BUZZWORD O REALTÀ?

Il termine Developer Experience si sta facendo largo sempre di più nelle conversazioni dell'industria IT. Spesso confuso per buzzword, la Developer Experience è tutt'altro che un trend passeggero. Al contrario, il concetto di Developer Experience sta dando forma ad un nuovo tipo di professionalità, nuove classi di funzionalità all'interno di prodotti per l'IT e, non da ultimo, sta ridefinendo la cultura e i processi aziendali. Altro che buzzword, verrebbe da dire. Stiamo parlando di un vero e proprio asset strategico per le organizzazioni. Non ci resta dunque che approfondire la DX attraverso questo white paper. Cosa aspettarsi da questa lettura?

Partendo da una definizione, esploreremo quali sono i pilastri della DX, impareremo così a conoscerla e a misurarla. Osserveremo gli scenari negativi dei contesti in cui l'esperienza degli sviluppatori non rientra tra le priorità aziendali, e poi, tutto a un tratto, verremo folgorati dai benefici che ottengono le organizzazioni che hanno riconosciuto l'importanza della DX. Infine, passeremo alla pratica, scopriremo come migliorare la Developer Experience in prodotti e organizzazioni, stando alla larga dalle insidie.

Questa lettura forse non farà di te un DX Expert - la nuova figura professionale che impareremo a conoscere - ma di certo ti darà gli strumenti per comprendere perché e come iniziare un percorso in questa direzione, nonché gli elementi per discuterne all'interno del tuo contesto di lavoro.

INDICE

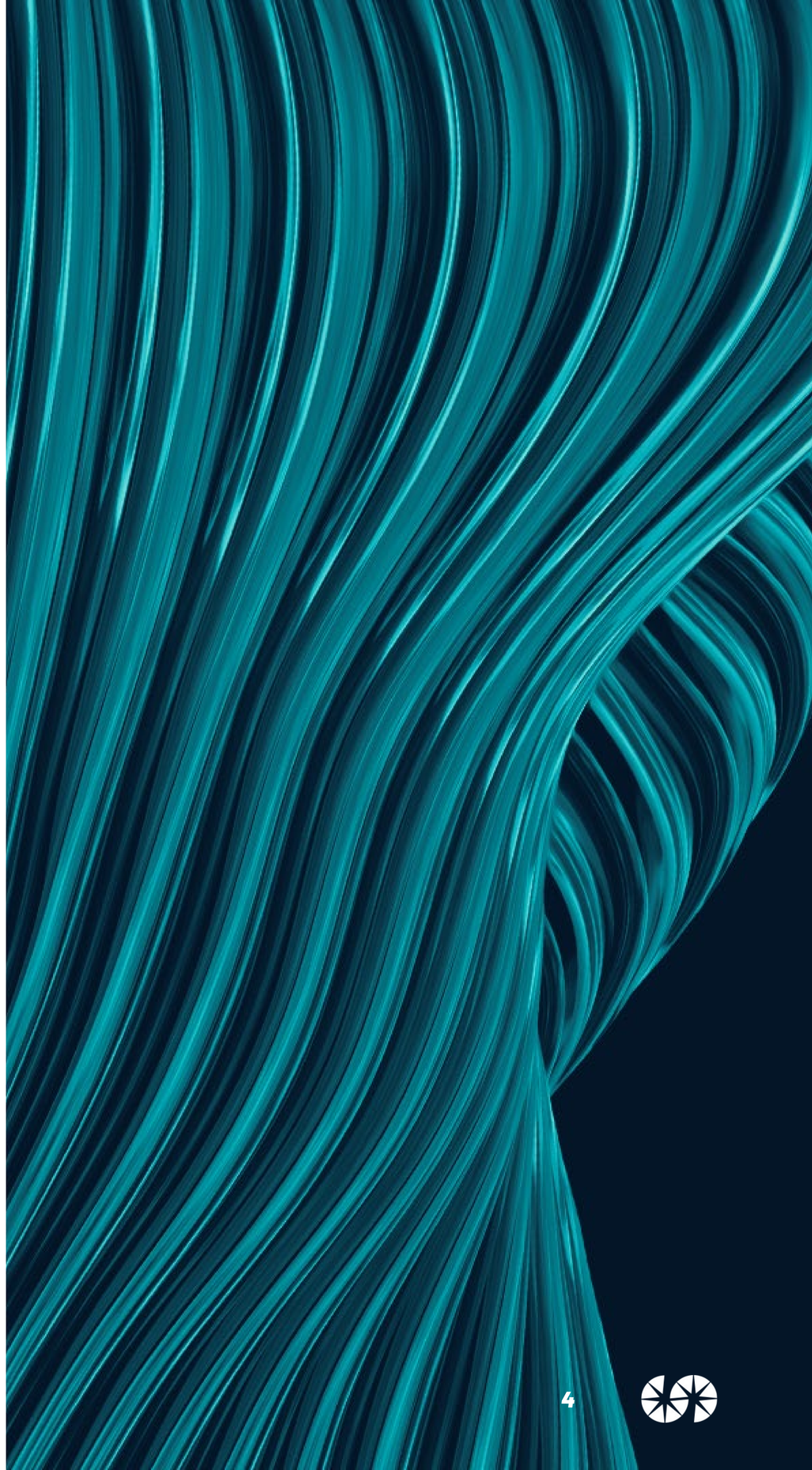
COS'È LA DEVELOPER EXPERIENCE?

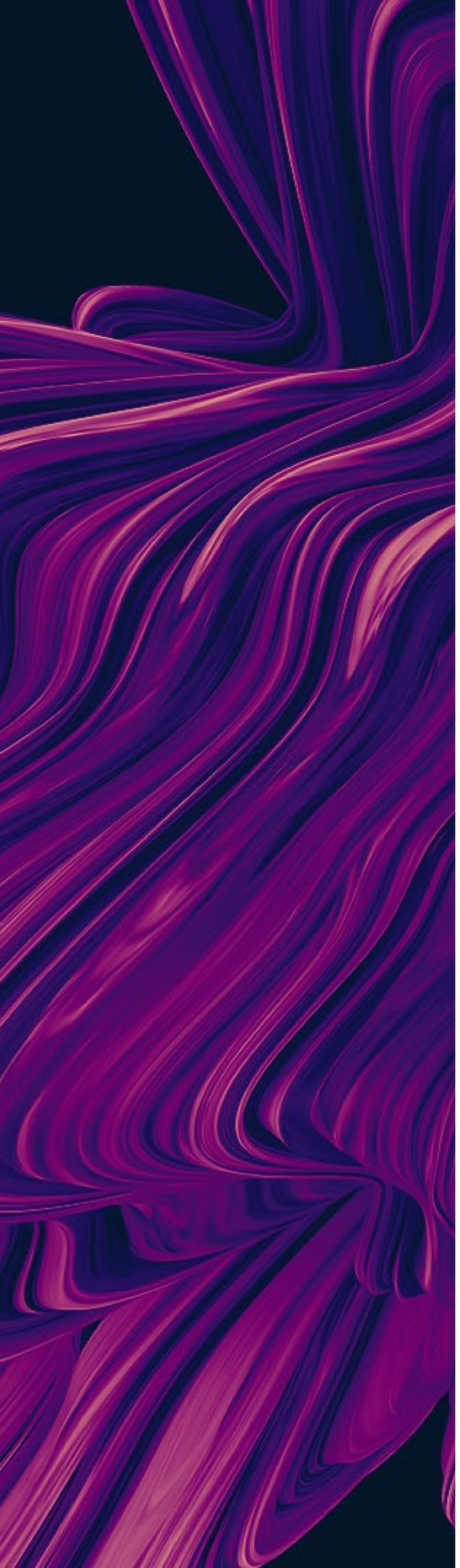
Il termine Developer Experience - in breve DevX o DX - può essere usato con accezioni diverse. Da un lato, possiamo inquadrare il tema di Developer Experience puramente da un punto di vista di human capital. In questo caso, siamo nell'ambito delle risorse umane e ci riferiamo al set di soft skill e processi di lavoro che vengono messi in gioco per migliorare la produttività di un team di sviluppatori e le dinamiche di collaborazione.

Allo stesso tempo, il termine DX sta anche ad indicare la capacità che hanno gli strumenti software - come ambienti di sviluppo e scripting - di semplificare il cosiddetto inner loop dello sviluppatore, ossia il suo flusso di lavoro, rendendolo di fatto più produttivo. La Developer Experience è quindi l'equivalente della User Experience, quando gli utenti primari del prodotto, della piattaforma o del processo di lavoro sono gli sviluppatori.

Volendo provare a dare una definizione di Developer Experience che metta d'accordo sia l'aspetto tecnologico, sia quello umano, possiamo dire che **la DX è l'insieme delle sensazioni che uno sviluppatore prova durante lo svolgimento dei suoi task**.

In ogni caso, capiremo che la Developer Experience è molto più di una bolla iridescente, destinata ad esplodere senza lasciare traccia. Sia nell'accezione tecnologica, sia in quella umana, la DX permette di avere un impatto su ciò che i team producono, contribuendo di fatto al raggiungimento degli obiettivi aziendali.





I QUATTRO PILASTRI DELLA DEVELOPER EXPERIENCE

La Developer Experience, così come ogni pratica che mira ad ottimizzare le modalità di lavoro, deve essere prima di tutto posizionata in un contesto organizzativo pronto a promuoverla e massimizzarne i benefici. I presupposti principali per una buona Developer Experience sono quattro.

ARCHITETTURA E CONTESTO ADEGUATI

È necessario bilanciare correttamente la complessità del contesto in cui si opera, sia quello organizzativo, sia quello architetturale. In questo modo è possibile garantire un ciclo di feedback corto che permette il miglioramento continuo.

STRUMENTAZIONE A SUPPORTO

In particolare, gli strumenti a supporto della DX sono quelli che automatizzano i task a basso impatto e valore aggiunto.

PROCESSI DOCUMENTATI

Una documentazione puntuale, come ad esempio delle checklist dettagliate, permette un'esecuzione precisa e standardizzata dei processi chiave.

CULTURA DEL TEAM

Abbiamo tutti bisogno di una motivazione e quella monetaria semplicemente non basta più. Bisogna trovare un purpose che faccia sentire le persone parte di qualcosa di più importante e su cui il loro lavoro ha un impatto diretto.

MISURARE LA DEVELOPER EXPERIENCE

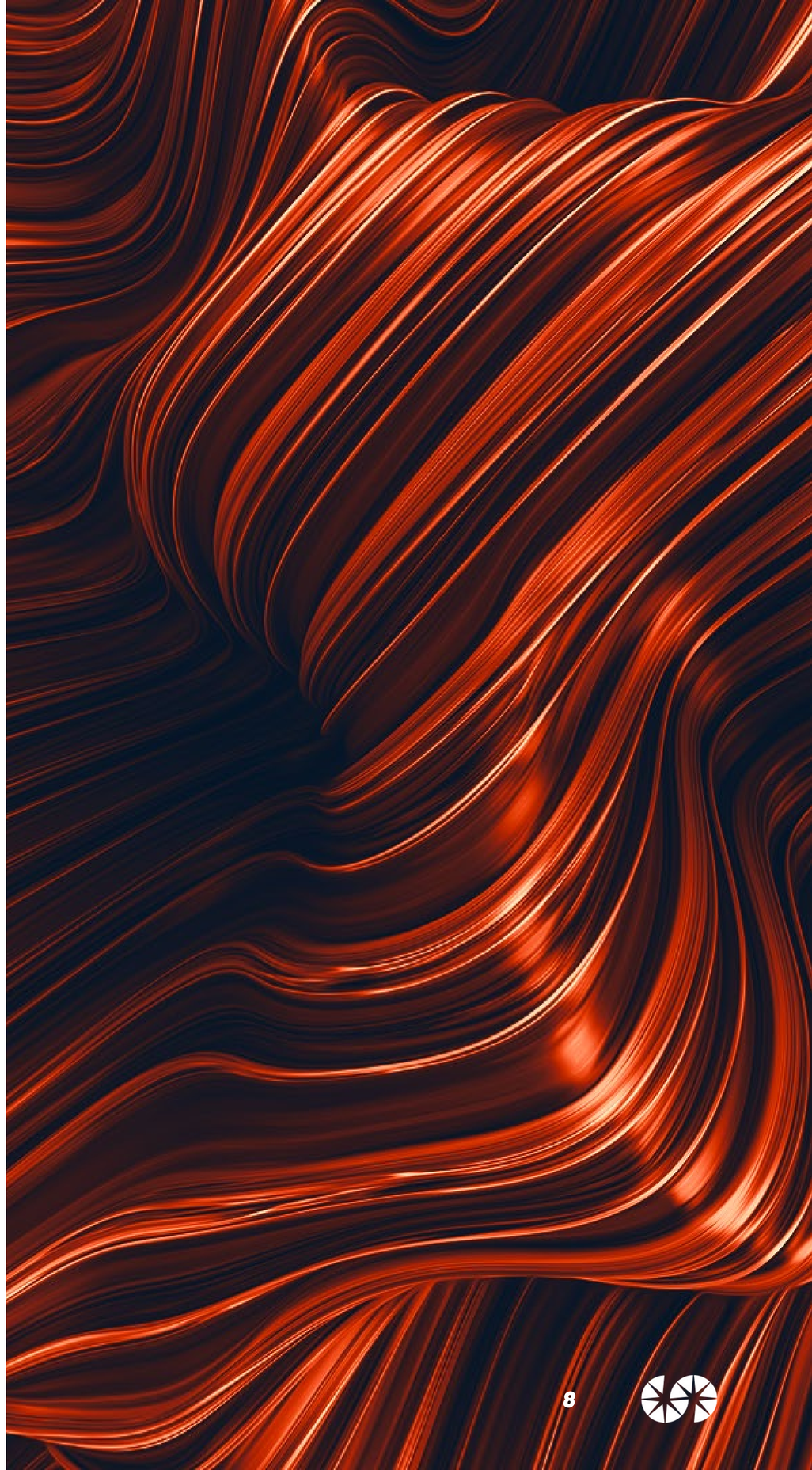
Ora che conosciamo la definizione e i pilastri della DX, sappiamo cosa dobbiamo misurare per valutare la Developer Experience, ovvero le emozioni e sensazioni che vanno a definire l'esperienza dello sviluppatore in quel determinato contesto di lavoro, o con quel determinato prodotto.

Possiamo quindi identificare tre KPI per valutare il livello di qualità della Developer Experience:

1. **Usabilità**: le funzionalità del prodotto, o il modo in cui un processo è strutturato, devono essere di semplice adozione per una persona nuova.

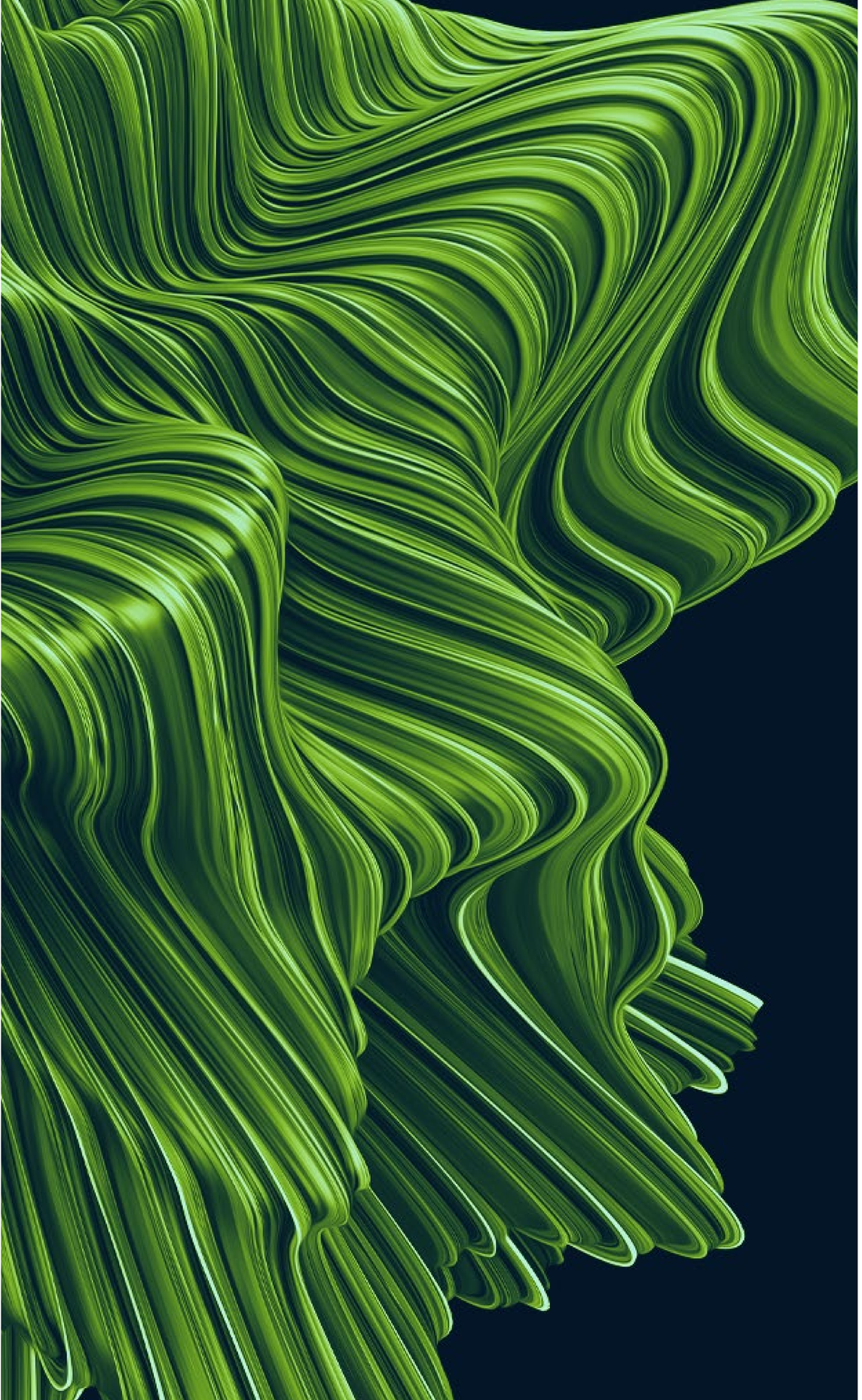
2. **Credibilità**: software e organizzazioni devono garantire ai propri utilizzatori e dipendenti una risposta alle loro domande e soluzioni ai loro problemi.

3. **Raggiungibilità**: la semplicità con cui un utente o un dipendente è in grado di raggiungere o identificare funzionalità, processi e persone di cui ha bisogno per un determinato task.



COME NASCE LA DEVELOPER EXPERIENCE?

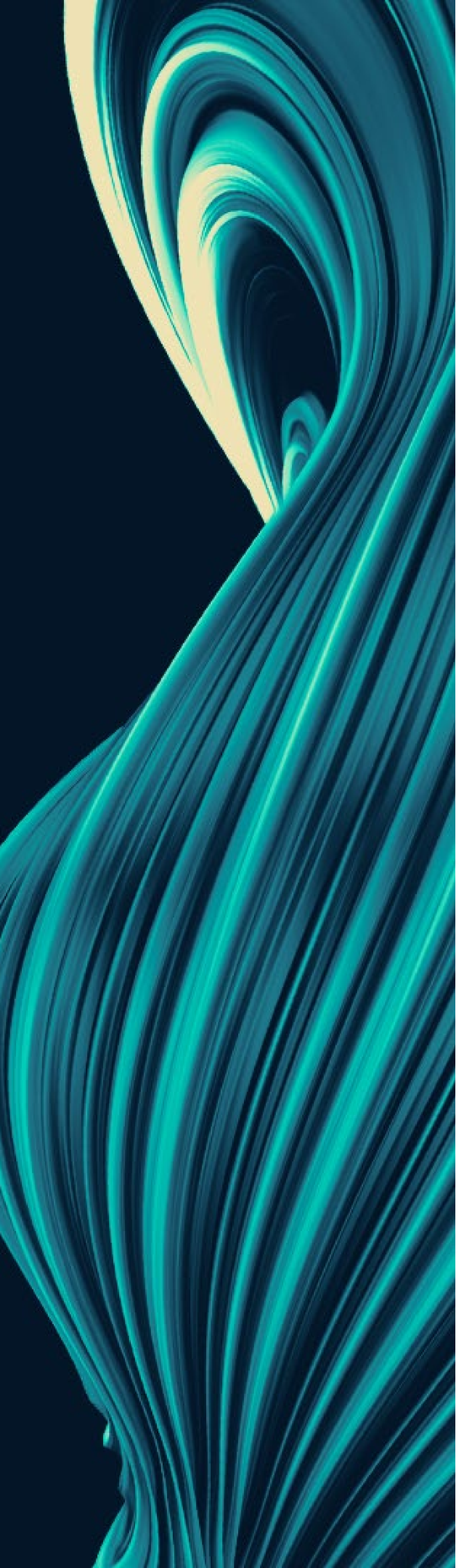
Nonostante sia un trend degli ultimi anni, La Developer Experience è la risposta ad una situazione di disagio che si protrae all'interno dell'industria del software da molto tempo. La crescente velocità con cui l'industria si muove, non ha fatto altro che peggiorare la situazione per sviluppatori e ingegneri, diventando di fatto insostenibile.



Per **RedMonk** - azienda nel settore IT e focalizzata su analisi di mercato - nonostante ci sia una pletora di strumenti, spesso eterogenei, che semplificano specifici task allo sviluppatore (dalla scrittura della prima riga di codice al rilascio in produzione), la varietà dei tool peggiora la situazione perché costituisce un ulteriore fattore di stress. Inoltre, non tutto può essere semplificato con un tool.

A pesare sia sulla produttività dello sviluppatore, sia sulla qualità del suo lavoro, è infatti anche la difficile comprensione delle codebase che, con il tempo, diventano sempre più complicate.

Quindi, anche se la Developer Experience ha radici lontane, negli ultimi anni il problema è peggiorato rapidamente. Da un lato la complessità non ha mai smesso di aumentare, dall'altro il mondo attuale richiede risposte, soluzioni e prodotti software con standard sempre più elevati e in tempi sempre più ridotti.



COSA SUCCEDE QUANDO LA DEVELOPER EXPERIENCE VIENE IGNORATA?

Il risvolto della situazione appena descritta non può che essere negativo, non solo sul piano dell'efficienza ma anche su quello emozionale. Una cattiva Developer Experience determina quindi un costo in termini di bassa produttività e anche un "**costo emotivo**". Come per tutti i costi, è importante per un'organizzazione cercare di ridurre al minimo il costo emotivo a carico dei team DevOps, prima che questo gli si ritorca contro.

Ecco, ad esempio, quali sono le situazione che aumentano il costo emotivo, in altre parole, cosa non fare:

- Sovraccaricare gli sviluppatori con **troppe informazioni** che causano difficoltà nell'organizzazione e nell'assimilazione dei concetti (carico cognitivo).

- Al contrario, dare **troppe poche informazioni** causa dubbi e incertezze sul modo di operare o sulle conseguenze di determinate attività.

- Applicare lo stesso approccio a macchia di leopardo instaurando una cultura del **"one size fits all"**.

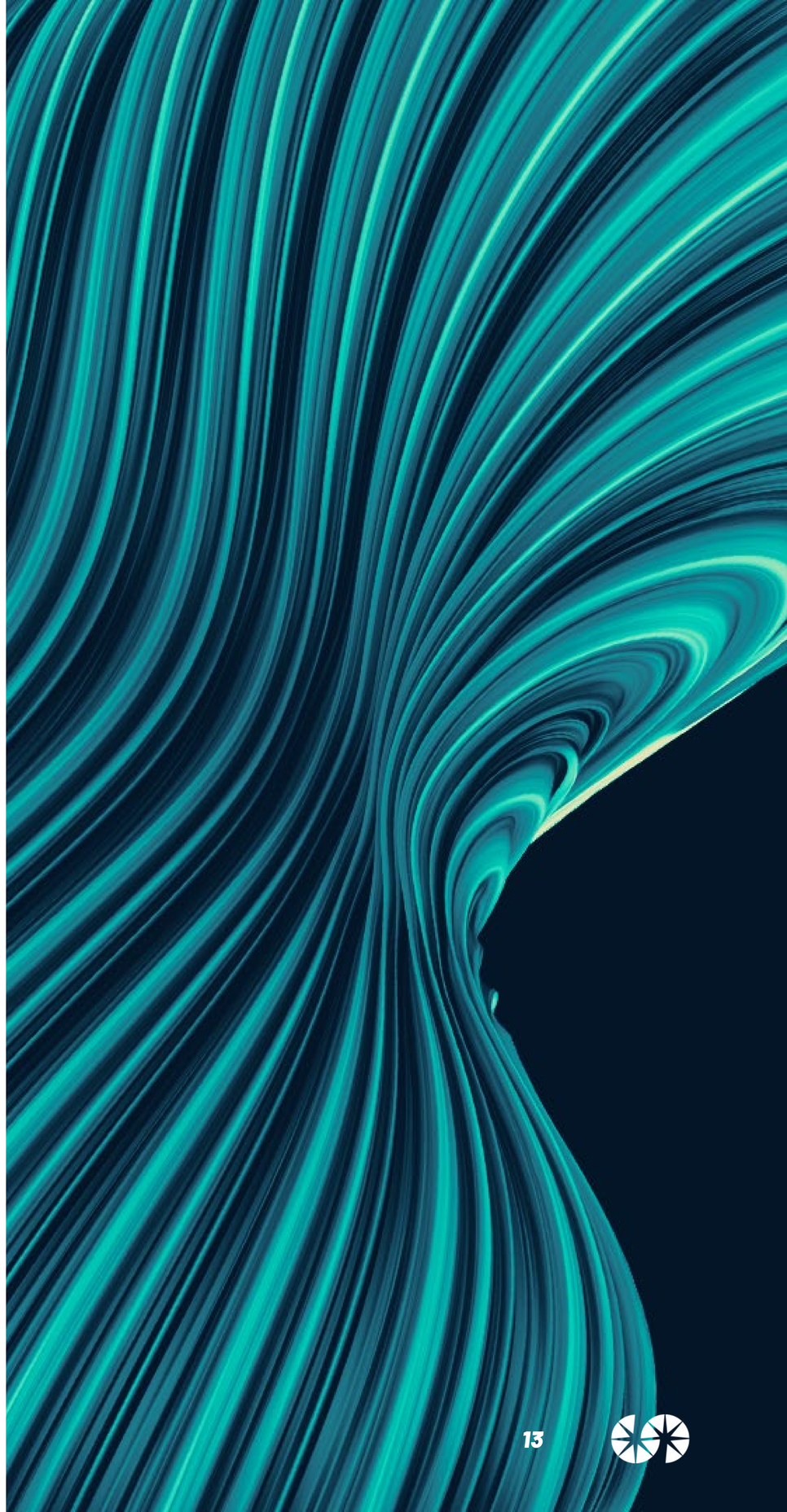
- **Sovraingenierizzare** attività semplici alla ricerca della sofisticatezza.

- Confondere l'Agile come un mezzo per scaricare i team dal loro lavoro, **aspettandosi che siano gli sviluppatori a fare tutto in autonomia**.

Il costo emotivo si collega ad un altro aspetto molto importante (dato il tasso di disruption dell'industria IT), ovvero l'impatto che la Developer Experience ha sul turnover dei dipendenti di un'azienda. Non supportare il benessere, la vena creativa, innovativa e l'indipendenza del lavoro del singolo sviluppatore può portare a disperdere conoscenze e competenze, invece che capitalizzarle, impattando in via definitiva sul prodotto.

C'è poi anche un altro elemento legato all'innovazione. **Forbes** pone in stretta relazione la qualità della User Experience di un determinato prodotto digitale con quella della Developer Experience di chi quel prodotto l'ha sviluppato. In altre parole, uno sviluppatore inefficiente, è uno sviluppatore che ha meno tempo di innovare e portare valore al prodotto.

Frustrazione emotiva, inefficienza, altissimo turn-over nei team di DevOps e prodotti meno innovativi: iniziamo a capire che la DX è più che un elemento di fregio per un'azienda ad alta connotazione digitale. È un aspetto strategico e come tale il suo posto è oggi in cima alla piramide delle priorità di ogni organizzazione che guardi al futuro.



I PROBLEMI CHE UNA BUONA DX RISOLVE

Analizziamo meglio i problemi che abbiamo appena citato. Guardandoli con la lente di ingrandimento, scopriremo che tra le pieghe dell'inefficienza e della frustrazione emotiva di un team si nascondono veri e propri mostri per la nostra organizzazione.

- **“Not My Problem” Mentality:** la tendenza dei componenti del team a non interessarsi alla risoluzione di una problematica che non rientra nella loro sfera di influenza.

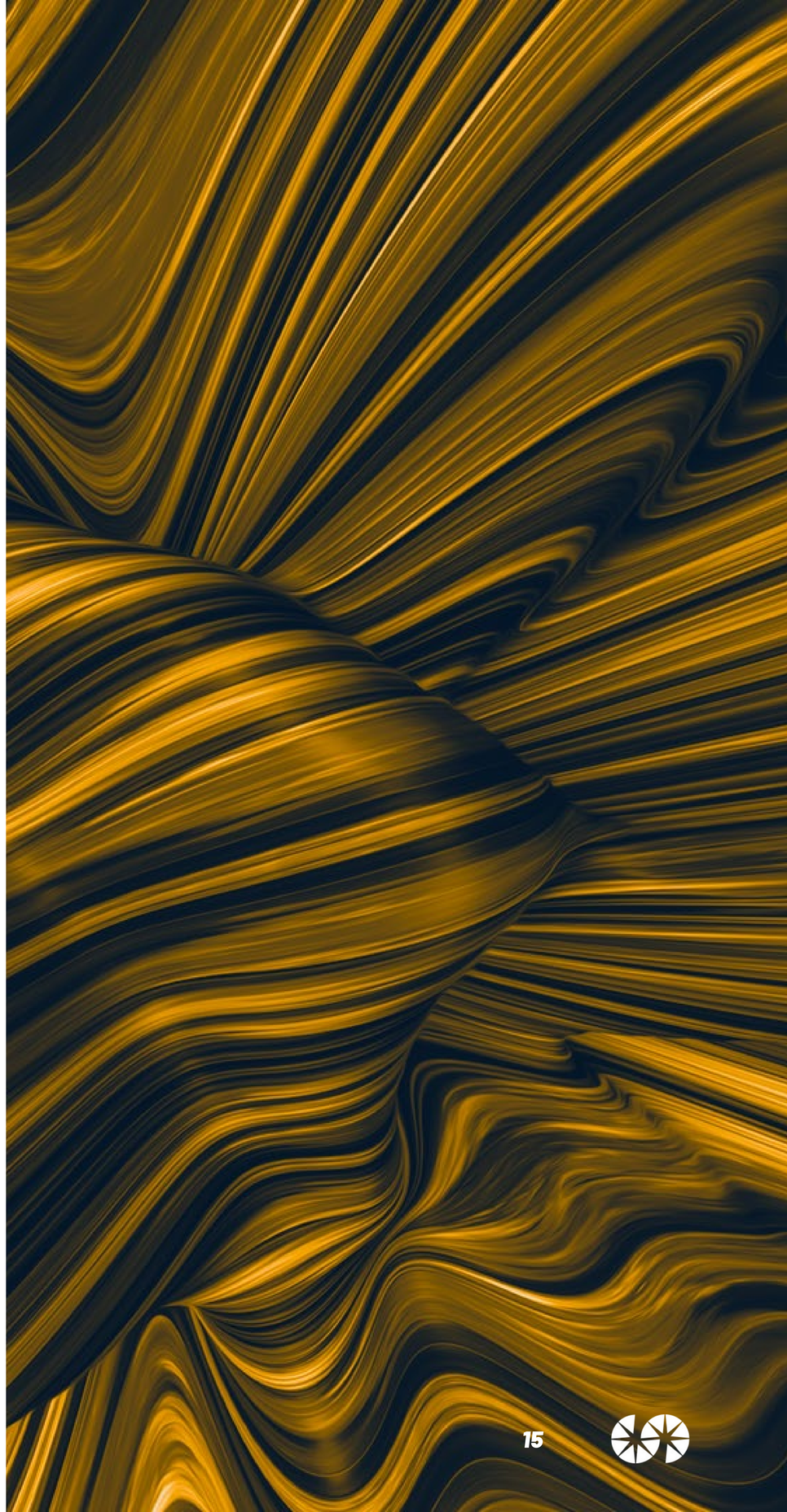
- **Toxic Team Culture:** ambiente dove rivalità e ambizione prendono il sopravvento rispetto agli obiettivi che si è prefissati di raggiungere.

- **Poor Code Quality:** poca cura nel rispetto delle best practice e nel garantire una bassa difettosità del prodotto.

- **Meaningless Work:** mancanza di visione e pianificazione a lungo termine che può portare il team a navigare a vista, portando avanti attività inutili rispetto all’obiettivo finale.

- **Demotivated Team:** il clima all’interno del team non permette ai suoi componenti di lavorare in maniera armoniosa, non favorisce la collaborazione e lo spirito di squadra.

- **Disconnect Between Business and IT:** l’organizzazione non riconosce il valore di investimenti a lungo termine nel reparto IT e pertanto non supporta e investe in nuove iniziative.



COME FAVORIRE UNA BUONA DEVELOPER EXPERIENCE

Ora che abbiamo ben chiaro il concetto di DX e la sua importanza, possiamo approfondire come implementarla all'interno della nostra organizzazione e quali sono gli approcci su cui il mercato si sta orientando. Possiamo identificarne cinque: grazie a questi step, potrai migliorare la DX all'interno del tuo contesto di lavoro, a beneficio della qualità, della produttività e del benessere lavorativo.

PRODUCT MENTALITY

Anche per le piattaforme e i prodotti pensati per sviluppatori è indispensabile applicare quelle che possiamo definire come “product mentality”. Solitamente, quando siamo alla ricerca di nuove opportunità per migliorare il nostro prodotto o lanciarne uno nuovo, andiamo ad analizzare paint points e bisogni non soddisfatti del target di mercato.

Sulla base di questi insights, si procede con lo sviluppo di un prodotto o di caratteristiche specifiche atte a soddisfare questi bisogni.
Beh, questo stesso approccio può e deve essere usato anche per prodotti o strumenti pensati per sviluppatori.

DEVELOPER JOURNEY

Comprendere il developer journey è cruciale per identificare limiti e aree di miglioramento. Si può iniziare prendendo parte ad attività “sul campo” e capire come gli sviluppatori vivono la loro quotidianità.

Questo ci permetterà di conoscere passo dopo passo cosa uno sviluppatore deve fare per essere operativo e portare del codice in produzione - dal setup iniziale della macchina, al rilascio in produzione, fino al supporto post produzione.

Spesso all’interno di questo processo ci sono molte attività manuali tediose che possono essere automatizzate per rendere tutto il procedimento, compreso l’onboarding all’interno del team, più veloce e semplice.

MINIMIZZARE IL FEEDBACK LOOP

Per attività eseguite molto spesso è importante minimizzare il feedback loop degli sviluppatori. Tim Cochran, nella sua ricerca sulla **developer effectiveness**, ha stilato una lista di processi che uno sviluppatore tipo è solito seguire all'interno della sua quotidianità e per ciascuno di essi ha stimato un tempo di "alta efficienza".

Questo tempo dovrebbe diventare il punto di riferimento oltre il quale l'operatività del team perde di efficienza.

Ad esempio, la root cause analysis di una problematica in uno scenario di alta efficienza non dovrebbe superare le ventiquattr'ore. Oppure, nello stesso scenario, il lancio di un nuovo servizio in produzione non dovrebbe superare i tre giorni di lavoro.

STIMOLARE LA COLLABORAZIONE

Eliminare i silos e instaurare una cultura aziendale basata sulla collaborazione non è facile, ma non è mai troppo tardi per incominciare!

Si può iniziare banalmente organizzando pause pranzo congiunte, in modo che le persone possano incontrarsi in modo informale.

È utile anche creare opportunità e spazio per condividere attivamente le informazioni, ad esempio attraverso conferenze interne o hackathon in cui i dipendenti di diversi team e dipartimenti possono incontrarsi e imparare gli uni dagli altri.

Il ricorso a standup giornaliere per colleghi di altri team è un'altra opzione per favorire la comprensione reciproca e creare empatia, soprattutto per i team che hanno forti dipendenze.

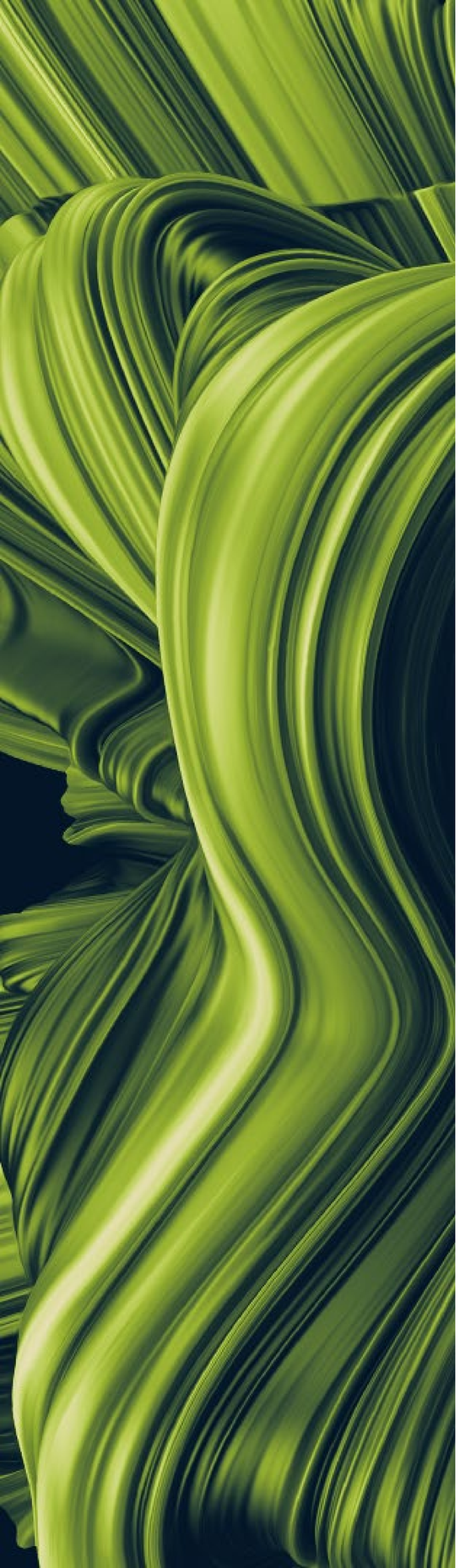
OPEN-INNOVATION E SPERIMENTAZIONE

Promuovere l'open-innovation e la sperimentazione può essere più complicato da implementare rispetto ad altri approcci, perché richiede maggior coinvolgimento da parte di tutta l'organizzazione.

Tuttavia, promuovere la sperimentazione permette a ciascun dipendente di liberarsi dal peso psicologico di un eventuale errore, di sentirsi di conseguenza libero di innovare, di essere propositivo e di esprimere in maniera aperta i propri pensieri con altri colleghi.

Questo ambiente favorisce un'innovazione che parte dal basso, che è in grado di portare beneficio al prodotto al pari di una direzione strategica pensata dalla leadership.

Su questo tema Google ha pubblicato una ricerca che indaga il modo in cui la stessa Google garantisce la **psychological safety** dei dipendenti, a vantaggio dei propri obiettivi aziendali.



IL FUTURO DELLA DEVELOPER EXPERIENCE

I cinque approcci che abbiamo appena visto andranno a definirsi meglio con il tempo, perché sempre più organizzazioni li adotteranno e di conseguenza ne miglioreranno gli aspetti che, ad oggi, possono sembrare più critici. Una pubblicazione di Red Monk intitolata **The Developer Experience Gap** prova a delineare quali saranno gli aggettivi che andranno a descrivere la prossima generazione di Developer Experience.

Anche questi sono elementi da tenere in considerazione quando progettiamo la DX dei nostri processi o prodotti:

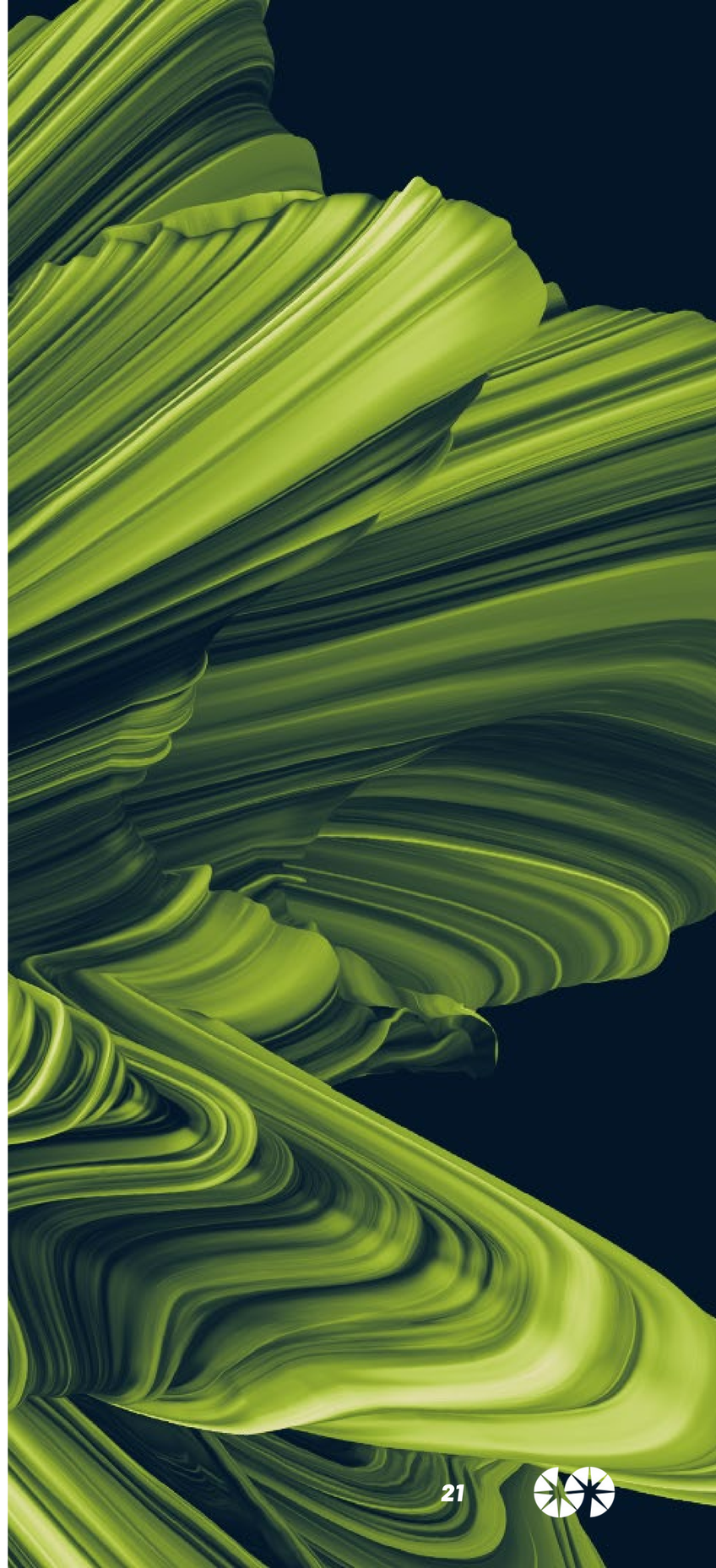
- **Comprensiva:** i fornitori di servizi evolveranno, andando oltre le loro aree di competenza di base. Estenderanno la loro base funzionale orizzontalmente, per offrire un'esperienza di sviluppo più completa e integrata.

- **Developer Native:** maggiore focus sul consentire agli sviluppatori di utilizzare gli strumenti con cui hanno più familiarità.

- **Elegante:** maggiore attenzione nelle modalità di fruizioni dei servizi da parte degli sviluppatori, per cercare di rendere più fluida l'esperienza di utilizzo.

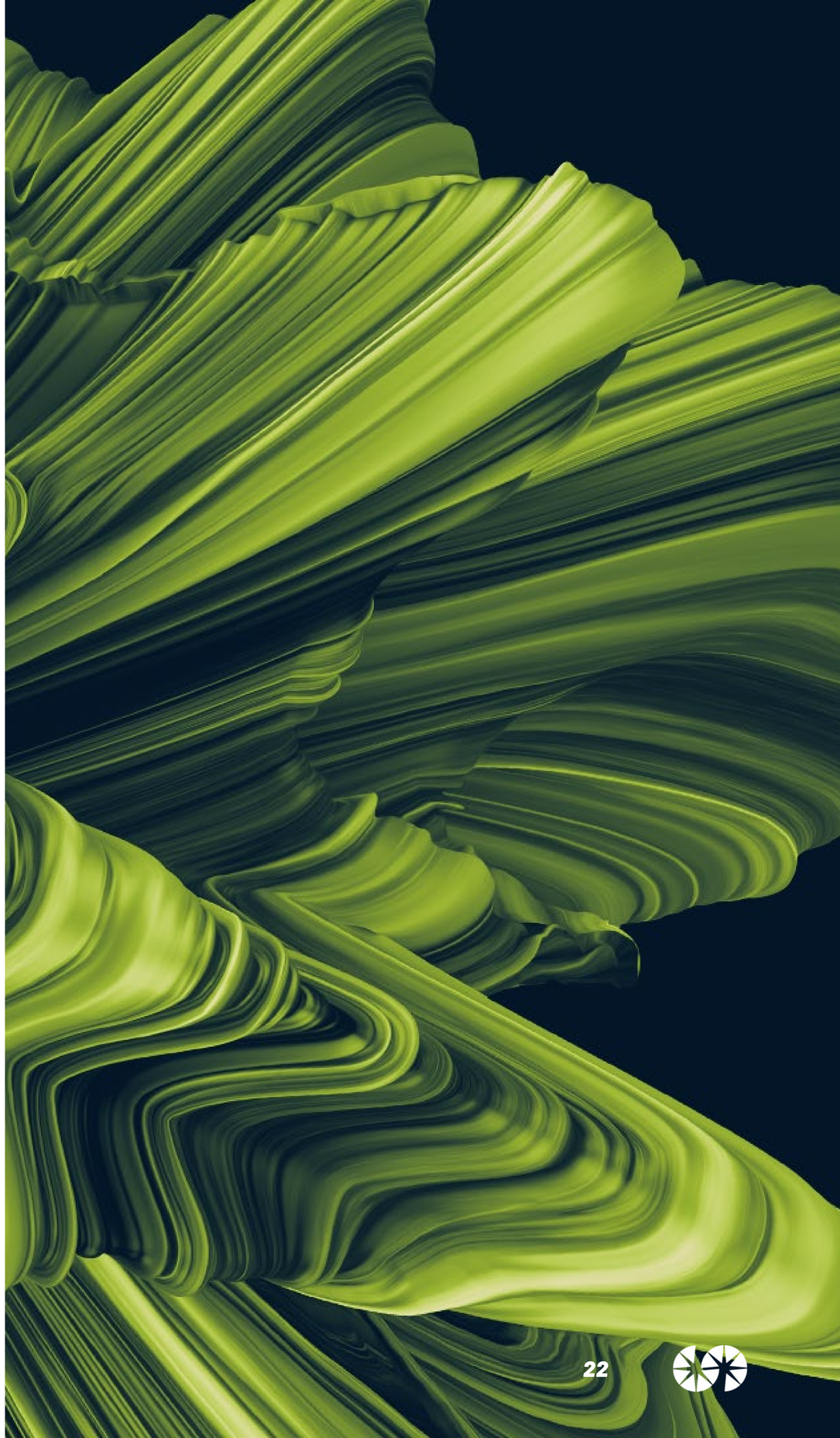
- **Multi-Runtime:** sempre meno limiti agli ambienti target e alle piattaforme che i sistemi integrati riusciranno ad indirizzate.

- **Multi-Vendor:** sempre meno verticalizzazione dell'offerta: il nuovo approccio sarà basato sull'ecosistema.



Cos'altro aspettarsi dal futuro? Sicuramente con l'evoluzione della Developer Experience evolveranno anche le professionalità a supporto.

Un primo esempio di questo fenomeno lo possiamo già osservare sul ruolo di **Platform Engineer** - il successore del Site Reliability Engineer - che ha come compito quello di supervisionare l'adozione delle Internal Developer Platform per rendere più semplice la vita degli sviluppatori nei team di prodotto.



IL RUOLO DELLE INTERNAL DEVELOPER PLATFORM

Abbiamo appena citato le Internal Developer Platform, o IDP, di cosa si tratta? Le Internal Developer Platform servono a unificare ed integrare tutti gli strumenti, in modo che i vari team possano operare in maniera self-service. Per questo motivo giocano un ruolo importante nella DX dei team di sviluppo.

L'utilizzo di una IDP permette al team di focalizzarsi sul valore delle attività, sgravando gli sviluppatori dalle complessità tecniche sottostanti, dalle operazioni manuali o di manutenzione. Per questo, le IDP hanno un impatto positivo sulla produttività e sul benessere del team di sviluppo.

L'utilizzo di una Internal Developer Platform permette inoltre di raccogliere metriche chiave in maniera strutturata, utili alla misurazione della Developer Experience, come: le prime impressioni del team, le prime esperienze di utilizzo e i primi successi. Tutti questi sono fattori molto importanti, infatti, non dobbiamo dimenticare che la Developer Experience non è uno standard. **Ogni realtà dovrebbe essere in grado di costruire la propria DX ideale, a misura del contesto in cui si trova.**

Un esempio di prodotto di mercato, nato dalle fucine di Spotify, che dà la possibilità di mettere in piedi una Internal Developer Platform è **Backstage**. Nel **Case Study sul Centro Medico Santagostino** - un'azienda italiana di successo che gestisce una rete di poliambulatori - raccontiamo come abbiamo realizzato una Internal Developer Platform per la digitalizzazione dell'Healthcare full-cloud, utilizzando proprio Backstage.

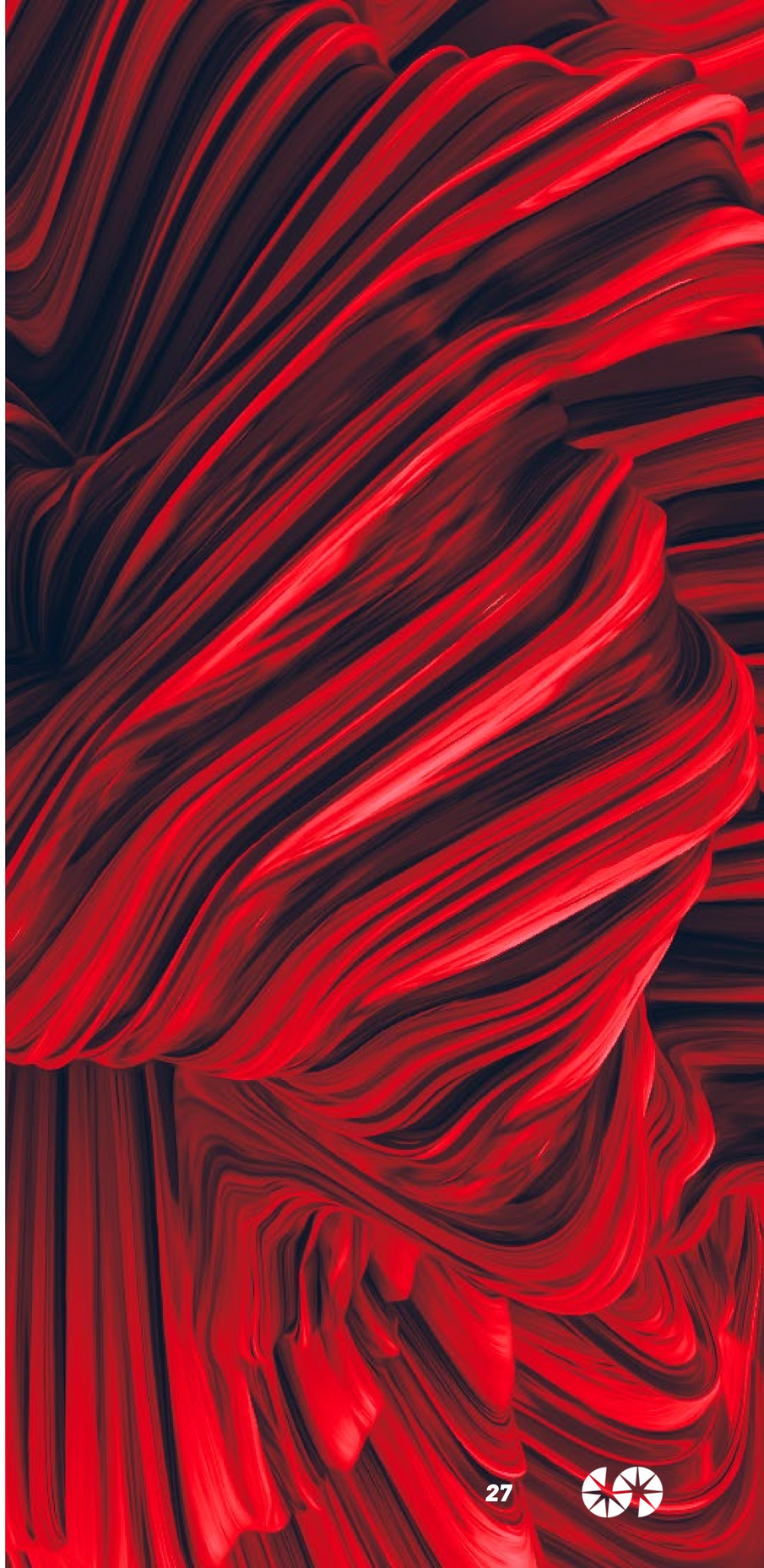
INSIDIE NEL PERCORSO VERSO UNA BUONA DEVELOPER EXPERIENCE

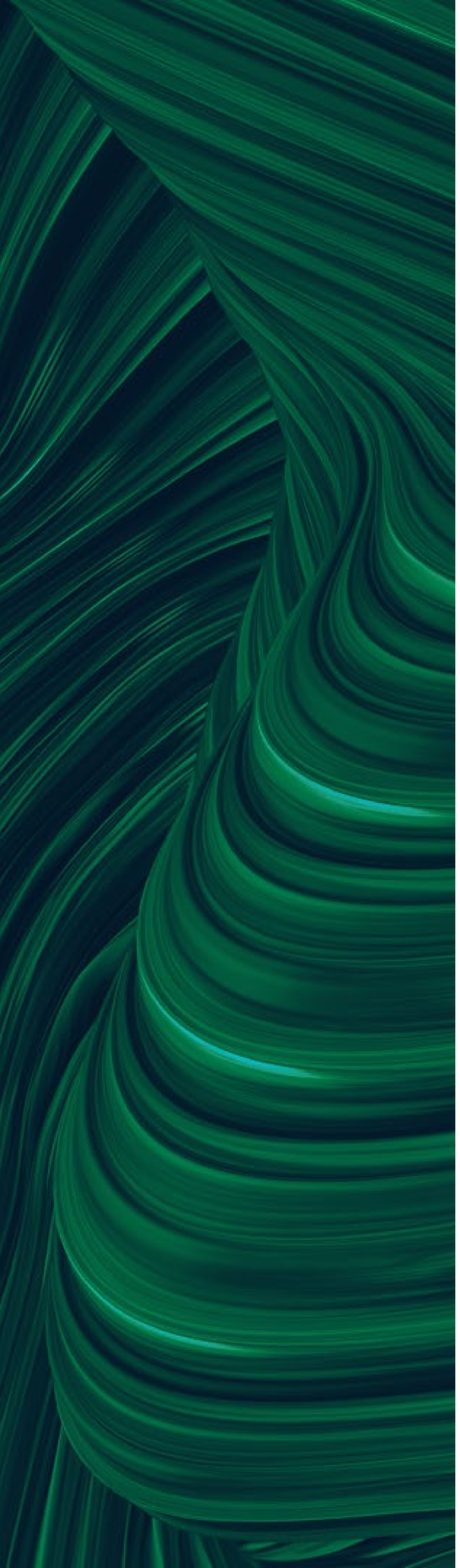
Man mano che le aziende crescono, i loro prodotti maturano così come le sfide che è necessario affrontare. Una di queste sfide è il precario **equilibrio tra sicurezza e usabilità**.
L'usabilità riguarda strettamente l'esperienza, sia essa dell'utente o dello sviluppatore. Tuttavia, spesso c'è un conflitto di priorità tra sicurezza e usabilità, tra gli sviluppatori del prodotto e quelli della piattaforma. In qualità di Platform Engineer, vuoi costruire una base sicura per il lavoro futuro.
Devi sviluppare una soluzione che sia di alta qualità, sicura ed estensibile.
I Product Engineer, però, sono spesso più vicini alle scadenze imposte dai clienti. Potrebbero dunque dover prendere scorciatoie su test e sicurezza per rispettare una data scadenza.

Un altro potenziale mal di testa tra i team sono le **vulnerabilità nei pacchetti di terze parti**.
Di solito, i team di sviluppo installeranno nuove librerie, probabilmente open source, per rendere più semplice l'implementazione di alcune funzionalità.
A seconda della struttura aziendale, tuttavia, potrebbe spettare al Platform Engineer o ai team di sicurezza correggere eventuali vulnerabilità rilevate in questi package esterni e applicare regole di export control. È importante trovare un meccanismo automatico e in co-responsabilità che indirizzi questo problema.

Questi esempi per dire che, a seconda del profilo con cui ci interfacciamo, è opportuno capire come bilanciare le necessità e le responsabilità di tutti. Ad esempio, gli sviluppatori tendono a fare del loro meglio quando lavorano su sistemi che apprezzano e trovano coinvolgenti. Migliorare l'esperienza degli sviluppatori di un sistema è un modo efficace per migliorare la qualità del codice e la soddisfazione dei dipendenti a lungo termine, ma non è opportuno farlo a scapito della sicurezza. Allo stesso modo, quando la sicurezza ha una priorità eccessiva, porta a sistemi sgradevoli per gli utenti e per gli sviluppatori che ci lavorano. Ciò può portare a uno sviluppo di funzionalità scadente e a un ritmo di cambiamento che si ferma. Trovare un equilibrio tra queste due priorità è complicato, ma cruciale per un prodotto e una DevX vincente.

In definitiva, piuttosto che vedere altri reparti o team come avversari, è fondamentale riconoscere che tutti fanno parte dello stesso team e che ciò che è buono per un team è probabilmente buono per l'intera azienda. Se ti interessa questo tema, ti consigliamo gli articoli che abbiamo redatto su **DevSecOps**, **Cloud DevSecOps** e il white paper gratuito **Guida alla Cloud Native Security**.





LA DEVELOPER EXPERIENCE PER L'INCLUSIONE E L'ACCESSIBILITÀ

Negli ultimi anni, l'industria IT - e in generale tutte le organizzazioni mondiali - si stanno impegnando sempre di più ad adottare comportamenti organizzativi che migliorino il livello di inclusione e diversità, andando a riformare il modo in cui la workforce collabora o il top management ragiona e opera.

Il collegamento con questi due temi e la Developer Experience viene quasi naturale, poiché pensare **un prodotto che sia intuitivo e favorisca la produttività di chi lo utilizza significa allo stesso tempo pensare ad un modo efficace per prevedere e abbattere barriere cognitive**. Purtroppo, i più moderni look & feel delle interfacce grafiche spesso non riescono ad accomodare le esigenze di tutti. Pensiamo ad esempio ad una UI minimale il cui modo di comunicare con l'utente medio è tramite overlay o testi a scomparsa: ecco, in questo caso il lavoro di un software di screen reading sarebbe difficile se non impossibile rendendo il software inutilizzabile per una categoria di sviluppatori.

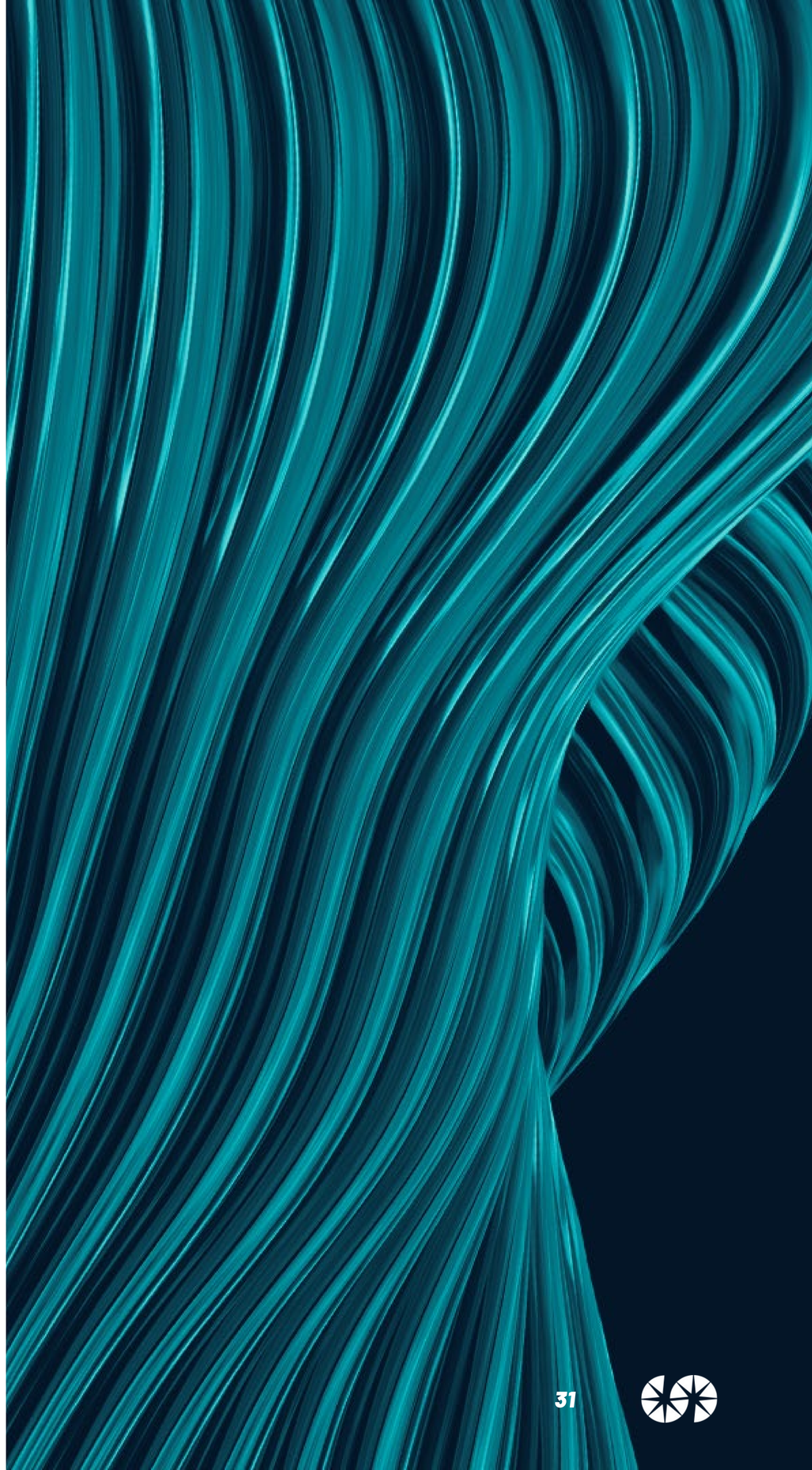
Sfortunatamente per le aziende, a volte un maggiore occhio di riguardo su aspetti come inclusione e diversità significa progettare UI, documentazioni e processi più articolati e, di conseguenza, più costosi da manutenere e realizzare. In conclusione, possiamo dire che inclusione, diversità e in generale accessibilità, nel contesto della Developer Experience si sostanziano nell'abilità di essere descrittivi il tanto che serve per permettere a quante più persone possibili di usare il nostro prodotto, o di consultare le nostre documentazioni. Bisogna quindi dare lo stesso livello di importanza all'eleganza che deriva dal completamento di una attività in maniera efficace ed efficiente e alla componente umana di ciascun utilizzatore.

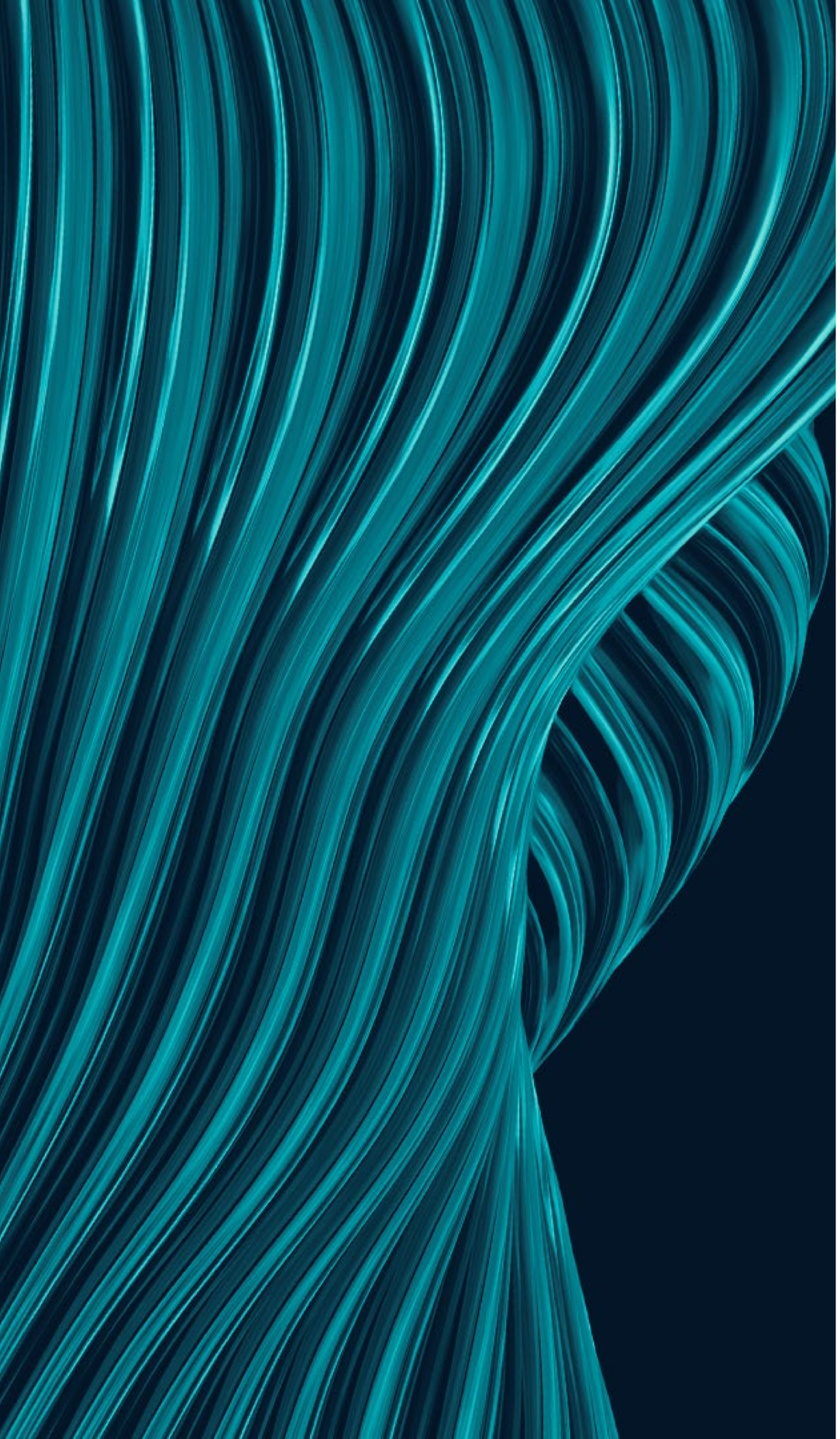
CLOUD NATIVE E DEVELOPER EXPERIENCE

Il paradigma di sviluppo Cloud Native ha cambiato il mondo della Developer Experience, dandogli una marcia in più. Così come il mondo Cloud Native, nemmeno la Developer Experience è monolitica. È possibile pensare e prevedere molte variazioni di una stessa esperienza di utilizzo di base, così come è possibile trovare vari tool all'interno del panorama Cloud Native che assolvono alla stessa funzione, ma in maniera diversa.

Possiamo ipotizzare che il driver principale di questa similitudine sia il paradigma dello shift-left, ovvero quello di spostare nelle mani dello sviluppatore sempre più titolarità e responsabilità dell'intero ciclo di vita del prodotto, partendo dalla scrittura del codice per arrivare al rilascio in produzione. Questa nuova mentalità Cloud Native si traduce in una migliore Developer Experience se il supporto arriva da ogni livello dell'organizzazione, soprattutto sul valore di un approccio shift-left.

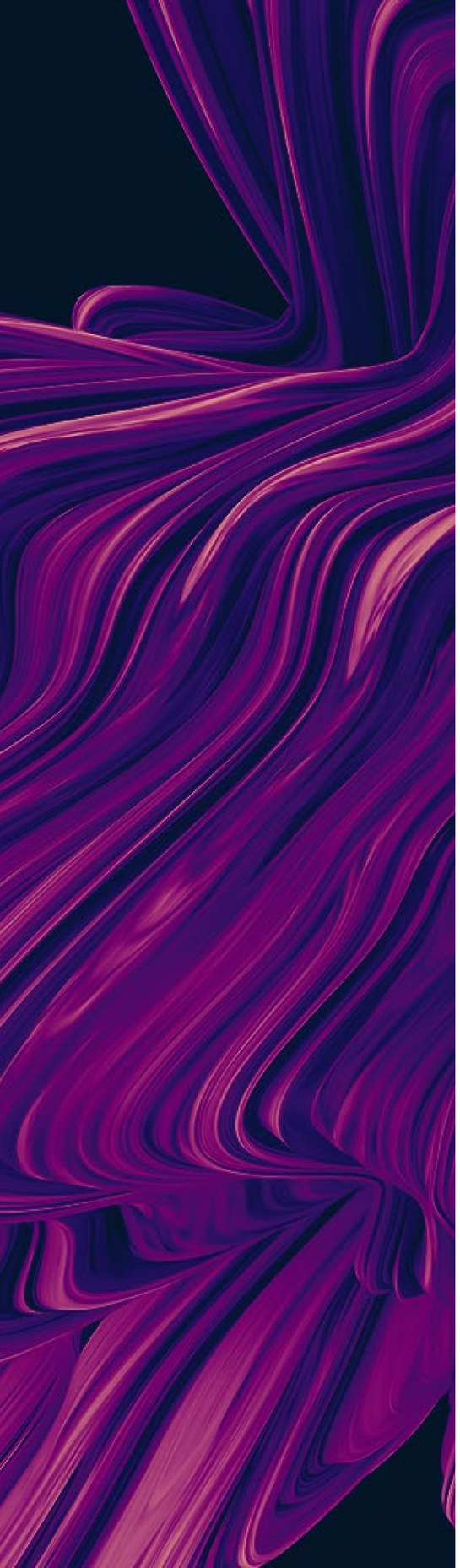
Questo maggior livello di coinvolgimento degli sviluppatori comporta anche l'instaurarsi di modelli di collaborazione. Gli sviluppatori lavorano a più stretto contatto con SRE e con i team di operation, concentrandosi maggiormente sugli obiettivi strategici relativi al prodotto e al rilascio del software e meno sulla lotta alle responsabilità e al finger pointing. Più i team lavorano insieme senza frizioni, più facile diventa il lavoro dello sviluppatore e maggiore sarà la sua propensione ad assumersi l'ownership dell'intero ciclo di vita. I diretti benefici di questo cambiamento sono: una diminuzione del time to market di nuove feature o miglioramenti al prodotto, la diminuzione del lead time e un aumento della qualità generale.





Un ottimo punto di partenza nel migliorare questi modelli di collaborazione è introdurre il concetto di "**Control Plane**" o **Centralized Developer Platform** all'interno della quotidianità dei team di sviluppo. Questo approccio permette di mediare le complessità delle attività che sconfinano dalle responsabilità delle varie personas (es: Developer, SRE, Platform Engineers), focalizzandosi nel dare uno strumento percepito affidabile. Questo è particolarmente utile non appena il lavoro dello sviluppatore sconfina nell'installazione o nel rilascio on Cloud del codice prodotto.

Si discute molto su quanto lo sviluppatore deve sapere o fare sul lato del deployment, ma la maggior parte degli esperti ritiene che gli sviluppatori devono solo potersi fidare del fatto che se forniscono il codice e la configurazione, la piattaforma eseguirà l'applicazione come specificato. In sostanza, questo Control Plane dovrebbe offrire agli sviluppatori un modo semplice per esprimere il loro desiderio per ciò che l'infrastruttura deve arrivare a rappresentare.



DEVELOPER PORTALS DI SUCCESSO

La Developer Experience, così come ogni pratica che mira ad ottimizzare le modalità di lavoro, deve essere prima di tutto posizionata in un contesto organizzativo pronto a promuoverla e massimizzarne i benefici. I presupposti principali per una buona Developer Experience sono quattro.

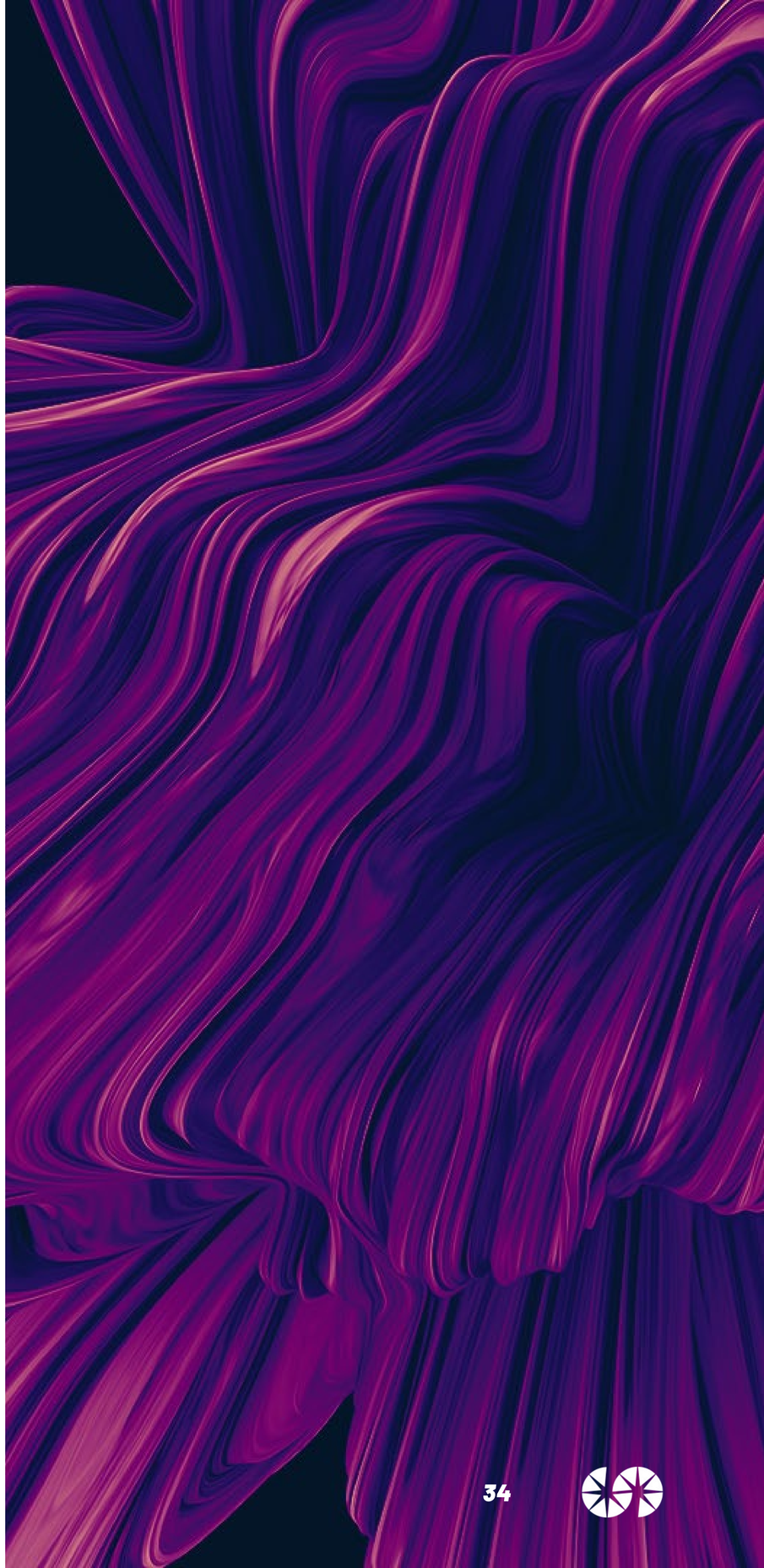
I Developer Portal hanno preso piede non appena i concetti di API Economy e API Platform sono stati assimilati dalle organizzazioni e sono così diventati la base per ogni servizio o piattaforma SaaS che si rispetti. C'è talmente tanta attenzione ed enfasi su questi portali che esiste anche un premio, i **DevPortal Awards**, che celebra il lavoro e l'impegno che le organizzazioni ci mettono nel rendere i loro Developer Portal funzionali, utili e rappresentativi del loro brand.

Riportiamo alcuni esempi di portali che, in alcuni casi, hanno vinto i DevPortal Awards e che sicuramente sono da considerare come best practice in materia di DX:

Visa Developer Portal: un eccellente esempio di portale per sviluppatori perché descrive in modo completo l'API e dimostra il suo utilizzo. Questo approccio "show, don't tell" può spesso portare a un'esperienza più pulita e più facile da digerire per gli sviluppatori. È necessario scavare di meno tra link e pagine per trovare elementi pertinenti nella documentazione.

Spotify for Developer: il principale vantaggio del portale per sviluppatori di Spotify è la forza della sua semplicità: è semplice e senza fronzoli. Ciò rende chiaro ciò che Spotify consente agli sviluppatori, e lo fa in un modo semplice e facile da navigare.

Plaid Docs: il portale per sviluppatori Plaid Docs è forse il portale più "dev-minded" in questo elenco. L'organizzazione è molto simile nel tema e nella struttura a come ci si aspetterebbe che funzioni un portale per sviluppatori. C'è un chiaro sistema di navigazione sulla sinistra e un generoso viewport con i contenuti sulla destra. Lo sviluppatore può facilmente prendere una API key nell'angolo in alto a destra della pagina e utilizzarla.



Wells Fargo Developer Gateway: Wells Fargo ha chiaramente progettato il proprio portale come un funnel, con elementi come il pulsante “Get started” e presentazioni di prodotti per caso d’uso e tipo di utente generico. Questa struttura aiuta a guidare i potenziali sviluppatori da una soluzione generale a un’implementazione più specifica.

Xero Developer: un ottimo esempio di come la presentazione dei contenuti sia importante tanto quanto il contenuto stesso. Sebbene il contenuto del portale Xero sia effettivamente efficace e ben organizzato, la natura del suo design è encomiabile. Xero ha fatto un investimento significativo nel suo brand e il loro portale lo rappresenta perfettamente.

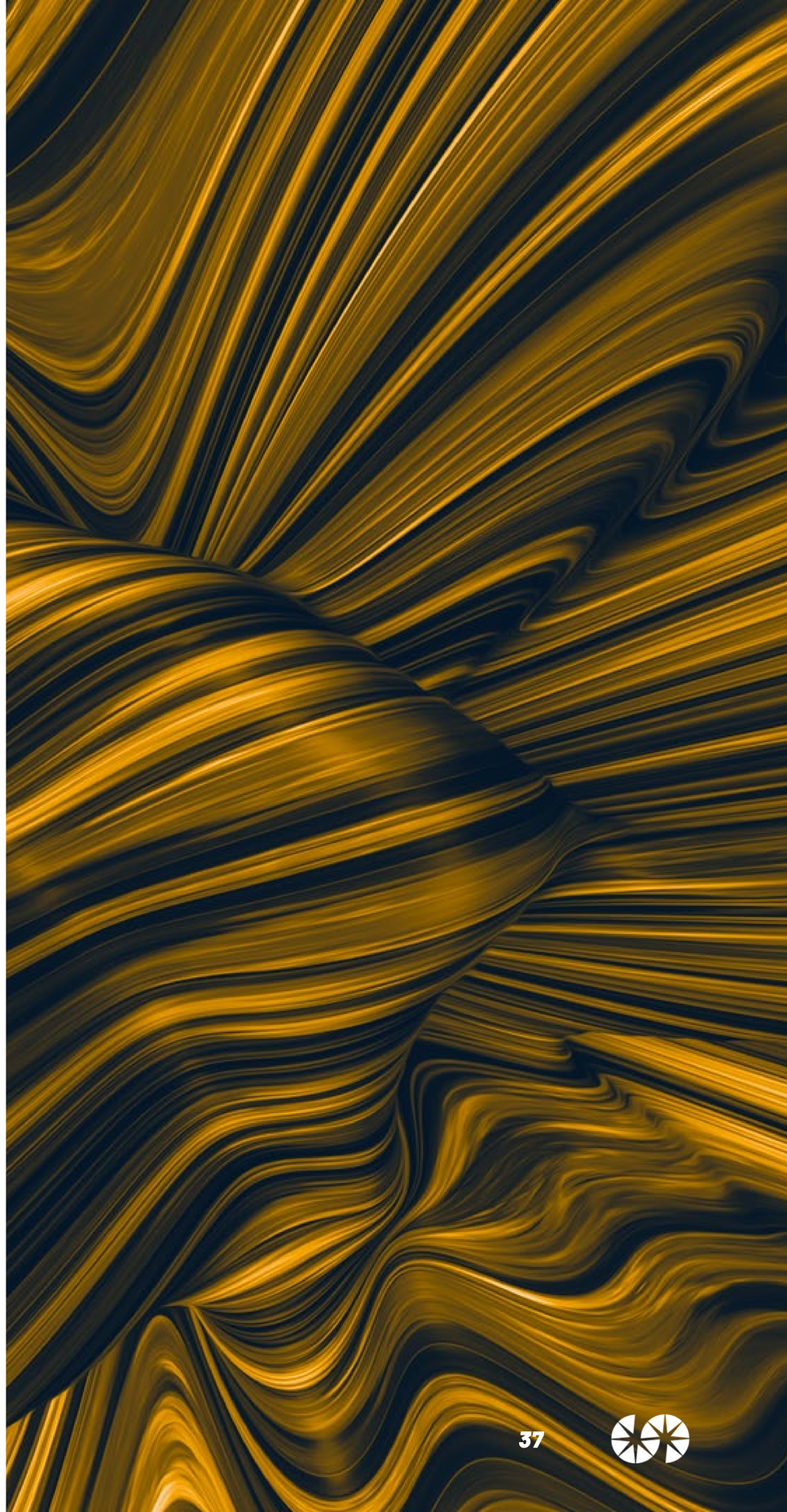
Twilio API Rerefence: Twilio è un eccellente esempio di un diverso tipo di portale per sviluppatori perché supporta un particolare tipo di consultazione e si sforza al massimo di farlo nel miglior modo possibile. Tutto ciò che uno sviluppatore potrebbe aver bisogno di sapere sull’ecosistema delle API di Twilio è chiaramente descritto nella pagina, con riepiloghi chiari per ogni elemento e ulteriori collegamenti che gli sviluppatori devono seguire per scoprire ulteriori informazioni, risorse e documentazione dell’API REST.

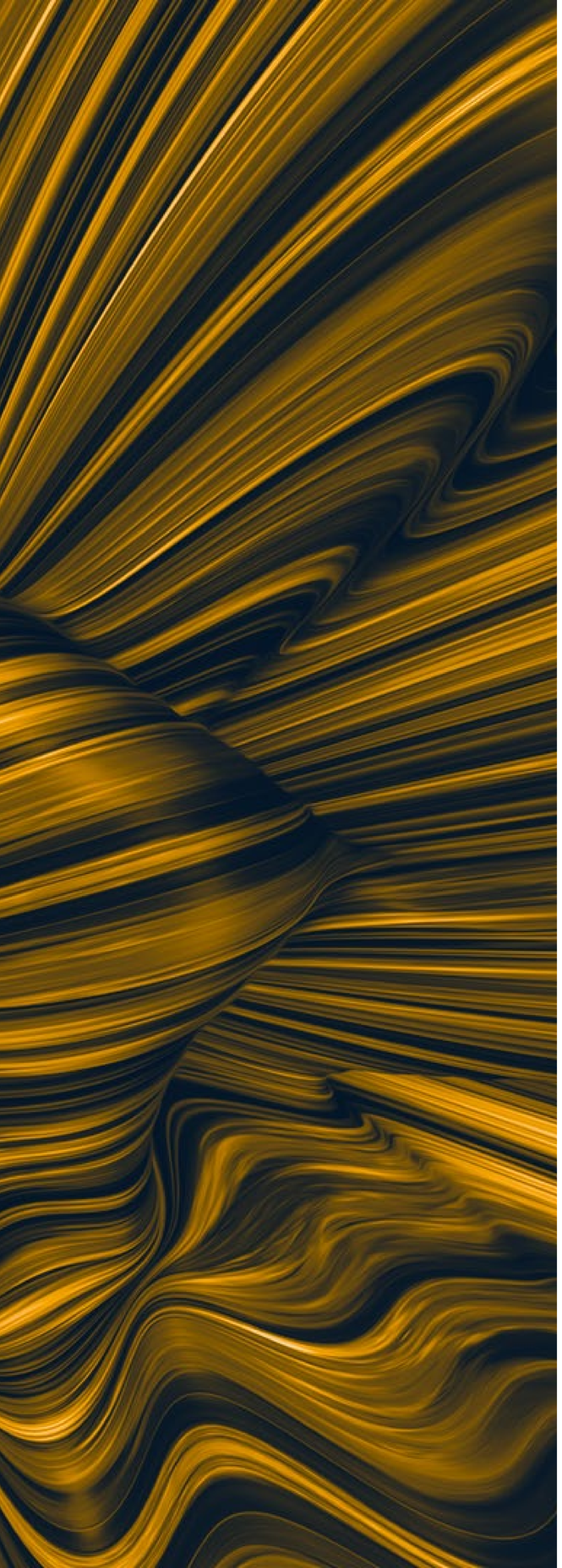
In conclusione possiamo dire che i Developer Portal sono fondamentali per lo sviluppatore-utente finale medio, in quanto fungono da punto di acquisizione di possibilità incalcolabili.

LA FIGURA DEL DEVELOPER EXPERIENCE EXPERT

Per le organizzazioni che hanno l'obiettivo di migliorare la Developer Experience dei propri prodotti è interessante sapere che nel panorama della workforce IT esiste nuova figura professionale dedicata alla DevX: il Developer Experience Expert. Come possiamo immaginare, il compito di questa persona è conseguire tutti gli obiettivi e i cambi organizzativi che abbiamo precedentemente descritto, ma non solo.

La posizione di Developer Experience agisce da ponte tra i developer teams e le funzioni corporate e amministrative, sfruttando un set di capacità interpersonali che gli permettono di dialogare efficacemente con entrambi i tipi di stakeholder.



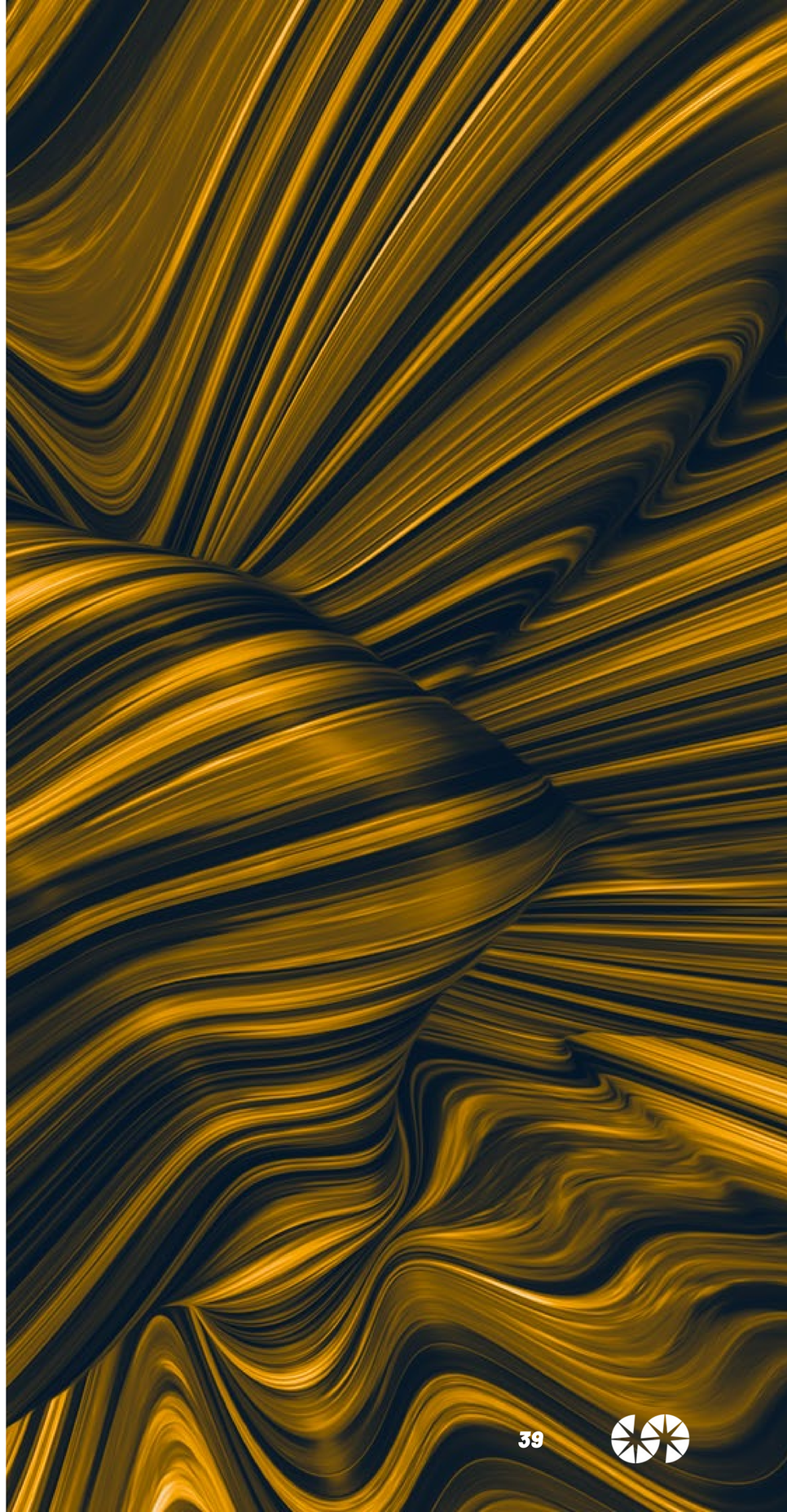


Come ci siamo detti, una buona Developer Experience include l'ottimizzazione dei set di strumenti e delle interfacce per gli sviluppatori. Ma per farlo, è necessaria una conoscenza approfondita del tipo di lavoro svolto dagli sviluppatori, nonché dell'ecosistema tecnologico in cui si svolge. Un esperto di DX avrà quindi bisogno di una profonda comprensione dei seguenti aspetti dell'ingegneria del software, di come vengono utilizzati e di cosa li rende efficaci:

- SDKs, APIs e librerie
- Developer tool chain, inclusi IDEs, text editors, compilatori, debugger e prompt
- Development framework
- Database, platform e strumenti associati
- Container orchestration platform (per esempio Kubernetes)
- Code snippet e tool per scaffolding
- Documentazione, includendo quella tecnica, coding standard e i tutorial più efficaci
- Ecosistema Open Source e Cloud Native

Oltre a queste capacità tecniche, come dicevamo pesano molto le soft skill. Il DX Expert deve avere la capacità di ascoltare, comunicare e imparare dalle esperienze degli altri, deve interfacciarsi con i livelli di management più elevati per procurarsi gli strumenti e gli ambienti giusti, adatti agli sviluppatori.

Chiaramente, è un lavoro che non andrà bene per tutti. In questo ruolo un background nello sviluppo è probabilmente essenziale e anche l'esperienza nella gestione dei team sarebbe molto vantaggiosa.

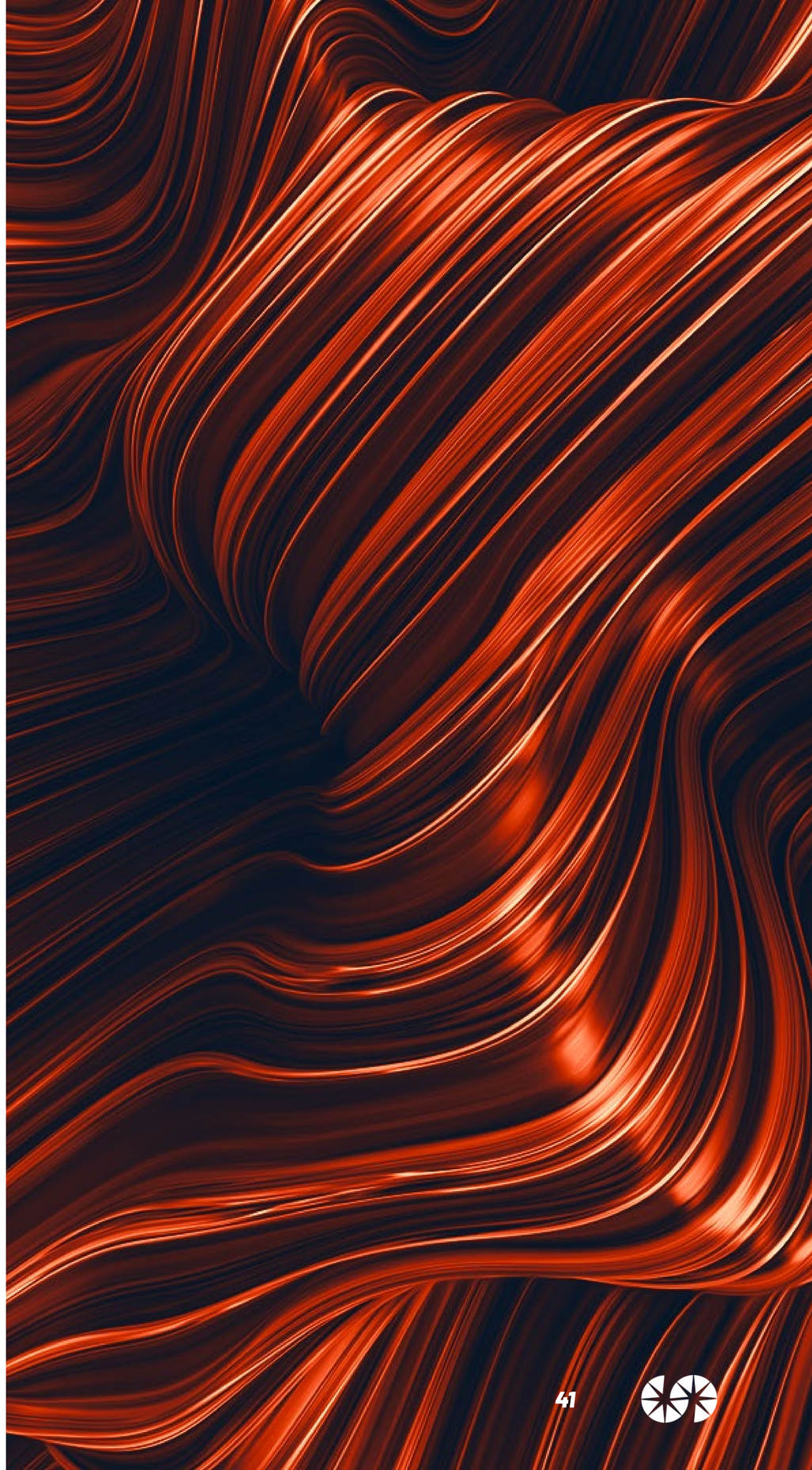


SPARKFABRIK E LA DEVELOPER EXPERIENCE

Per SparkFabrik la Developer Experience è un tema fondamentale nonché parte integrante della nostra cultura aziendale. Non siamo solo Cloud e DevOps Engineer, Strategist, Designer e Cloud Native Expert: siamo noi stessi sviluppatori appassionati.

Ecco perché curiamo la DX dei nostri processi: per garantire un lavoro efficiente, innovativo e di qualità. Lo facciamo per i nostri team, ma lo facciamo perché questo inneschi un meccanismo virtuoso esternamente: progettiamo soluzioni software che aiutino a massimizzare la Developer Experience dei nostri clienti, in modo tale che anche la User Experience dei loro clienti possa a sua volta essere incrementata. Una catena valoriale win-win per tutti gli stakeholder, insomma.

Vuoi approfondire? **Ascolta la puntata di Let's talk about DevX di Codemotion con SparkFabrik**. Paolo Mainardi e Francesco Benigno - rispettivamente CTO e Software Developer di SparkFabrik - raccontano come hanno iniziato con la DevX e come la mettono in pratica quotidianamente.



CONCLUSIONI

Per concludere, ripassiamo velocemente i pilastri della Developer Experience che abbiamo visto all'inizio, applicandoli a SparkFabrik:

1. **Architettura e contesto adeguati:** il contesto organizzativo di SparkFabrik è agile, in questo modo possiamo garantire un ciclo di feedback corto: migliorare di continuo e rispondere così puntualmente alle mutevoli necessità dei mercati e dei clienti.

2. **Strumentazione a supporto e Processi documentati:** Il nostro workflow è trasparente e documentato nel **Company PlayBook**, dove riportiamo anche tutti gli strumenti al servizio dei nostri sviluppatori. Il lavoro è altamente standardizzato, a beneficio della serenità degli sviluppatori e della qualità dell'output finale.

3. **Cultura del team:** Il nostro è un settore altamente competitivo, poniamo costantemente l'accento sulla cultura aziendale per guadagnarci la fiducia costante delle nostre risorse umane, che sono l'asset più prezioso. Un assaggio della cultura di SparkFabrik la puoi trovare nel nostro **manifesto**.

Se ti interessa conoscerci ancora meglio, puoi sempre **unirti al nostro team**, o iniziare un progetto insieme a noi, come hanno fatto **loro**. In entrambi i casi, faremo grandi cose insieme (grazie anche alla Developer Experience).

SPARKFABRIK