



docebo



AI Won't Fix Your Developer Experience

Se scriviamo codice sempre più velocemente, perché consegnare software resta così complicato e la DevEx non va di pari passo?

Riccardo Galimberti

VP of Engineering



Alex Pagnoni

Developer Experience
Engineering Manager



Developer Experience, nella pratica



Una buona Developer Experience



Strumenti e infrastruttura

Ambienti ripetibili, test e CI/CD affidabili

Processi e autonomia

Obiettivi, responsabilità e accessi chiari

Cultura e leadership

Fiducia, supporto e feedback costruttivo

Architettura e codice

Modifiche comprensibili e verificabili

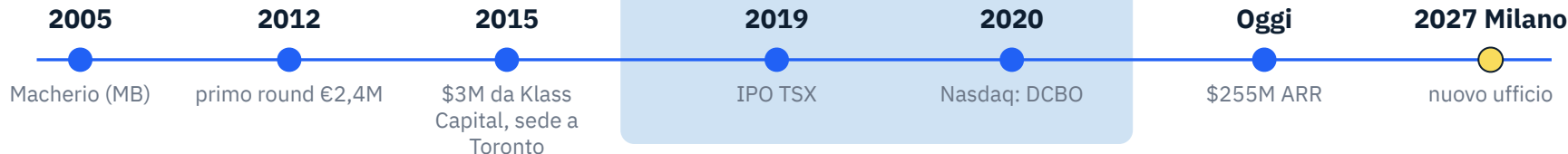
È l'esperienza complessiva. Difficile da ottenere al 100% in tutte le aree, ma ci si lavora



IL CONTESTO DOCEBO

Una learning platform... da Biassono al Nasdaq

HYPERGROWTH 2016–2022 • ARR DA \$11M A \$129M



500 milioni di chiamate alle nostre API al giorno
circa 5.800 al secondo, ogni secondo

AZIENDA

\$255M

ricavi ricorrenti annui

3.400+

clienti

900+

persone

8 paesi

su 4 continenti

INGEGNERIA

~230ingegneri, 35+ team
merge request in 12 mesi**30.000+****~60 / mese**

rilasci: core + servizi autonomi

Ci usano **Booking.com, Ubisoft, Tripadvisor, Zoom** • Se sei seduto lì, con buona probabilità ci hai usato anche tu senza saperlo

Più velocità, ancora attrito

87%

L'AI aiuta a consegnare
il lavoro più velocemente

72%

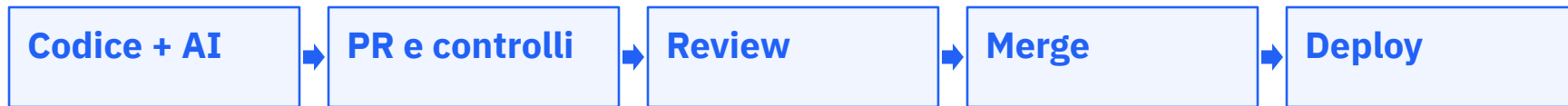
Attriti invariati o maggiori
rispetto a prima dell'AI

Dove finisce il tempo che abbiamo risparmiato?

Atlassian, State of Developer Experience 2026



Velocità del codice e velocità della delivery



Il guadagno locale

Scrivere una modifica più rapidamente

Le condizioni

Contesto, ambienti, review e test affidabili

Il rischio

Più lavoro in ingresso, capacità invariata

Paradosso: codice più veloce, delivery no. L'AI può rendere più visibili le attese nel resto del percorso.



Developer & Agentic Experience

Developer

Repository condiviso

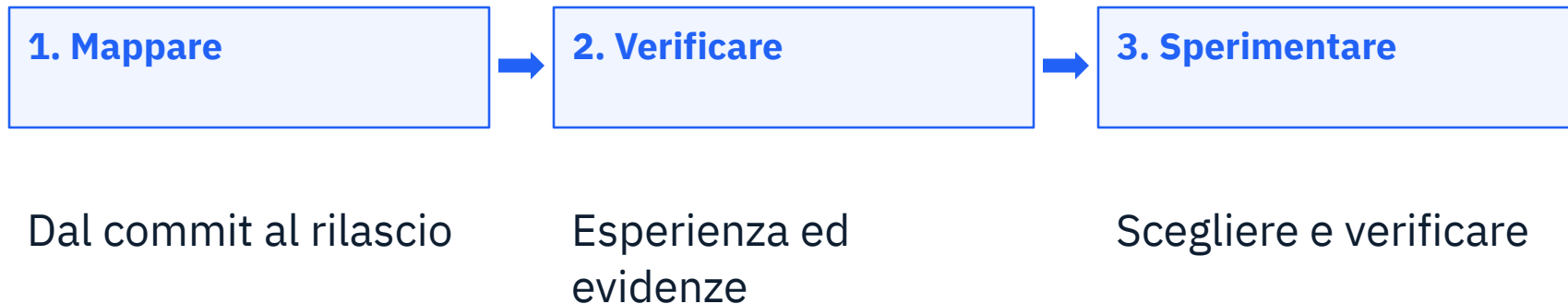
Agente



DevEx e Agentic Experience vanno sempre più di pari passo



Tre passi per migliorare la DevEx



**Non serve partire da una trasformazione enorme.
Una journey, un'ipotesi verificabile, un esperimento**



Uno specchio, non una pagella

Flusso

PR integrate / settimana e lead time al deploy

Attese

Review pickup e coda CI, p50 e p90

Qualità

Change failure rate e rilavorazioni dopo review

Esperienza

Sforzo, comprensione, context switching, carico cognitivo e tempo di verifica AI

Telemetria e ascolto: survey, interviste, osservazione del lavoro



Developer Lifecycle Map



Un percorso condiviso, con iterazioni e ritorni



Developer Journey Map

From first day to long-term impact. A great developer experience powers a greater Docebo.

STAGE	1. JOIN Become part of Docebo	2. SET UP Get a working environment	3. UNDERSTAND Learn the big picture	4. DEVELOP Build and make changes	5. VALIDATE Ensure quality and safety	6. DELIVER Get to production	7. OPERATE Run and support	8. EVOLVE Improve for the future
DEVELOPER GOAL	Feel welcome, informed and ready to contribute.	Have a smooth, secure and reliable development environment.	Understand the product, architecture, domain and ways of working.	Implement changes efficiently and with confidence.	Verify that my changes work, are safe and meet quality standards.	Get my changes reviewed, merged and deployed to users.	Monitor, troubleshoot and keep the system healthy.	Continuously improve the product, the codebase and my skills.
KEY QUESTIONS	<ul style="list-style-type: none"> What's my team? How do we work? Where do I find what I need? 	<ul style="list-style-type: none"> How do I get access? How do I set up my env? Is everything working? 	<ul style="list-style-type: none"> How does the system work? Where is the code? What are the key concepts? 	<ul style="list-style-type: none"> Where do I start? How do I make this change? Who can help if I'm stuck? 	<ul style="list-style-type: none"> How do I test my changes? Are there automated tests? Will this break something? 	<ul style="list-style-type: none"> How do I get a review? What's the deployment process? When will it be in prod? 	<ul style="list-style-type: none"> How do I monitor it? What do I do if something breaks? Where are the logs? 	<ul style="list-style-type: none"> How can I improve this? What's the roadmap? How can I contribute more?
TYPICAL ACTIVITIES	Complete onboarding, meet the team, learn processes and culture.	Request access, configure tools, set up local env, run first build.	Explore docs, read code, learn architecture, understand domain and user needs.	Pick a task, understand code, implement changes, run locally.	Run tests, linting, security checks, peer review, validate in staging.	Open PR, address feedback, merge, deploy via CI/CD.	Monitor metrics, check alerts, investigate issues, fix problems.	Refactor, reduce technical debt, propose improvements, learn new skills, contribute to standards.
KEY TOUCHPOINTS & TOOLS	HR, People Team, Onboarding Portal, Slack, Welcome Docs	Laptop, Identity & Access, Dev Environment, IDE, Internal Tooling	Engineering Handbook, Architecture Docs, Codebase (GitHub), Domain Resources	IDE, GitHub, Local Env, Feature Branches, Internal Libraries	Automated Tests, CI, Code Review, Static Analysis, Security Scans, Staging	GitHub, CI/CD, Release Process, Change Management	Monitoring, Logging, Incident Tools, On-call, Runbooks	Roadmap, RFCs, Tech Demos, Learning Resources, Communities of Practice
DEVELOPER EXPERIENCE	 Excited	 Hopeful	 Curious	 Focused	 Anxious	 Relieved	 Stressed	 Motivated
COMMON PAIN POINTS	Slow onboarding, unclear information, limited initial context.	Access delays, complex setup, environment issues, missing documentation.	Complex architecture, scattered knowledge, outdated docs, hard to find experts.	Large codebase, technical debt, unclear ownership, slow builds.	Flaky tests, long feedback times, unclear failures, environment inconsistencies.	Slow reviews, CI failures, complex release process.	Hard to find information, noisy alerts, unclear responsibilities.	Difficulty driving change, legacy constraints, lack of time, unclear prioritization.
OPPORTUNITIES (DEVEX)	Structured onboarding, mentorship, clear first-week plan, welcome experience.	Self-service provisioning, better automation, pre-configured environments, troubleshooting guides.	Unified documentation, service catalog, architecture visualizations, searchable knowledge base.	Faster builds, better tooling, clearer ownership, code navigation, internal SDKs.	More reliable tests, faster feedback, clearer guidelines, test environments on demand.	Streamlined review workflow, reliable CI/CD, visibility on deployment status.	Better observability, runbooks, incident tooling, blameless learning.	Support for refactoring, technical debt reduction, learning programs, hackathons, time for innovation.

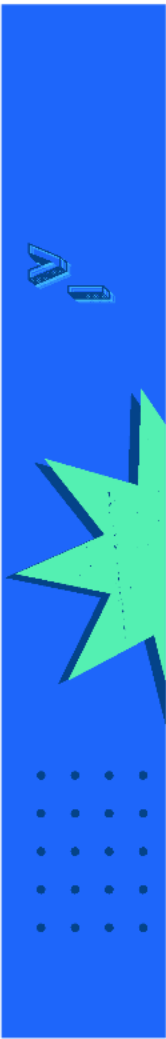
A BETTER DEVELOPER EXPERIENCE BUILDS A STRONGER DOCEBO

Developer Journey Local Development Setup

From a laptop to a running environment. A smooth setup unlocks developer productivity.

STAGE	1. PREPARE Get what I need	2. GET ACCESS Access to systems	3. SET UP Install and configure	4. BUILD & RUN Get it working	5. VALIDATE Ensure it's correct	6. BE PRODUCTIVE Start developing	OUTCOME Ready to contribute
GOAL What I want to achieve	Understand what I need and get my laptop ready.	Get the required access to repositories, tools and environments.	Install and configure the development environment.	Successfully build and run the application locally.	Verify that everything works as expected.	Start working on real tasks with confidence.	I can develop, test and contribute effectively in Docebo.
DEVELOPER ACTIONS What I do	<ul style="list-style-type: none"> Read the onboarding guide Check hardware requirements Prepare my laptop Complete mandatory training 	<ul style="list-style-type: none"> Request accounts and access Set up SSO / VPN if needed Join relevant Slack/Teams channels Get added to repositories 	<ul style="list-style-type: none"> Install required tools (IDE, SDKs, runtimes, DBs) Configure environment Clone repositories Set up local services (e.g. Docker, Kubernetes) 	<ul style="list-style-type: none"> Install dependencies Build the project Run the application locally Seed test data if needed 	<ul style="list-style-type: none"> Run automated tests Check core features work Verify integrations (e.g. DB, services, APIs) 	<ul style="list-style-type: none"> Explore the codebase Pick my first task Make a small change Run the full dev workflow 	I have a fully working local environment and I'm ready to contribute to the product.
TOUCHPOINTS & TOOLS Where I interact	<ul style="list-style-type: none"> Onboarding portal Hardware guide Welcome docs 	<ul style="list-style-type: none"> Okta / SSO GitHub / GitLab Slack / Teams Access request system 	<ul style="list-style-type: none"> IDE (e.g. VS Code) Docker / Kubernetes Language runtimes Internal setup scripts Dev environment config 	<ul style="list-style-type: none"> Build tools (e.g. Maven, npm) Local services (DB, Redis, etc.) Makefile / scripts Feature flags 	<ul style="list-style-type: none"> Test frameworks Local test data Quality tools (linters, SAST) API clients (e.g. Postman) 	<ul style="list-style-type: none"> IDE Internal documentation Code search Issue tracker (e.g. Jira) 	<ul style="list-style-type: none"> Working local environment Connected to the team First contribution
KEY QUESTIONS What's on my mind	<ul style="list-style-type: none"> What do I need to get started? Is my laptop compatible? Where can I find the setup guide? 	<ul style="list-style-type: none"> How do I get access? Who can approve it? Do I need a VPN? How long will it take? 	<ul style="list-style-type: none"> What tools do I need? Are there automated scripts? Where are the configuration files? Is there a standard setup? 	<ul style="list-style-type: none"> How do I build the project? How do I run it locally? Do I need specific data? What should I see when it's working? 	<ul style="list-style-type: none"> How do I run the tests? Which tests are required? How do I verify everything is working correctly? 	<ul style="list-style-type: none"> What should I work on first? Where can I find help? How do I follow best practices? Who can I ask if I get stuck? 	<ul style="list-style-type: none"> Am I fully set up? Can I work independently? Do I know where to find ongoing support?
COMMON PAIN POINTS What can go wrong	<ul style="list-style-type: none"> Unclear requirements Missing information Outdated documentation 	<ul style="list-style-type: none"> Delays in account provisioning Lack of access to repositories Unclear approval process 	<ul style="list-style-type: none"> Complex setup process Environment inconsistencies Version conflicts Missing dependencies 	<ul style="list-style-type: none"> Build failures Services not starting Hard to debug issues Slow build times 	<ul style="list-style-type: none"> Flaky tests Missing test data Environment differences Unclear what "working" means 	<ul style="list-style-type: none"> Hard to find relevant code Lack of context Unclear next steps First task too complex 	<ul style="list-style-type: none"> Still dependent on others Recurring setup issues Not feeling productive yet
DEVELOPER EXPERIENCE How it feels	Neutral	Frustrated	Frustrated	Challenged	Uncertain	Hopeful	Confident
OPPORTUNITIES (DEVEX) How we can help	<ul style="list-style-type: none"> Clear onboarding checklist Up-to-date hardware specs Short "day 1" video guide 	<ul style="list-style-type: none"> Automated access provisioning Clear approval flow Access status visibility Pre-configured groups 	<ul style="list-style-type: none"> One-click setup scripts Standardized dev containers Better error messages Environment health check 	<ul style="list-style-type: none"> Faster builds (caching) Improved local services Better debugging tools Troubleshooting guide 	<ul style="list-style-type: none"> Reliable and fast tests Consistent test data Clear validation checklist Pre-configured quality tools 	<ul style="list-style-type: none"> Guided first task Codebase walkthrough Mentorship / buddy system Context in documentation 	<ul style="list-style-type: none"> Measure time to productivity Continuously improve based on feedback
SUCCESS METRICS How we know it works	<ul style="list-style-type: none"> % of new hires with checklist completed before day 1 	<ul style="list-style-type: none"> Median time to get all access % of access requests automated 	<ul style="list-style-type: none"> Median environment setup time % successful setup without manual help 	<ul style="list-style-type: none"> % successful local build on first attempt Median time to run locally 	<ul style="list-style-type: none"> % passing tests on first run Time to validate environment 	<ul style="list-style-type: none"> Time to first PR Developer satisfaction (setup experience) 	<ul style="list-style-type: none"> Time to productivity Retention and engagement Positive feedback from new developers

REMOVE FRICTION. UNLOCK POTENTIAL.



«Dove sono finiti i soldi dell'AI?»

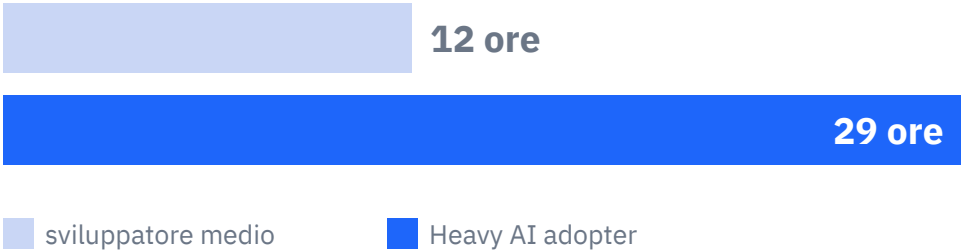
In Docebo ci interessa come lavorano i team, non solo quanto velocemente consegnano

In attesa di code review.

PR mergiate a settimana

3,5 vs 4,1

Ore per arrivare al merge

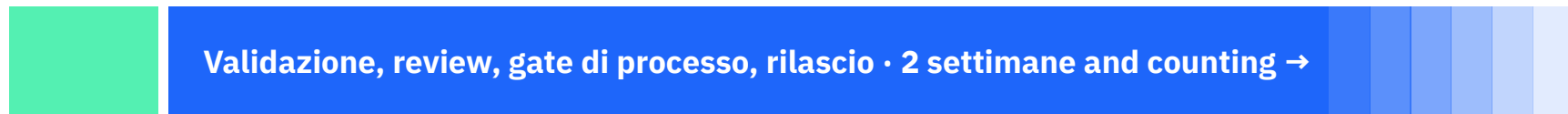


Dato: Sviluppatori Docebo, valori mediani delle ultime 12 settimane. Confronto tra gruppi, non un prima/dopo. Tempo lordo.

UN CASO DOCEBO

Dove il lavoro va in coda

Stima: ~~24~~ giorni → ~~12,5~~ giorni



Scrittura con agenti

< 2 giorni · ~\$90

Il coding è passato da 24 giorni a 2. Il resto no.



IL VINCOLO SI È SPOSTATO

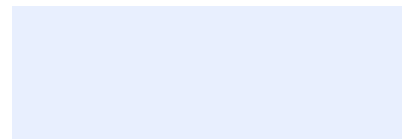
Il vincolo nuovo: chi verifica

Discovery

Codice

Verifica

Rilascio



la spec deve essere vera, non
plausibile

due persone, con nome e cognome
· test che provano il
comportamento voluto

«Un'ora guadagnata dove non c'è il vincolo è un miraggio.»

Eliyahu M. Goldratt



Cosa stiamo cambiando

Discovery

Ready to Build: la spec entra in sviluppo solo se validata, altrimenti torna indietro

Codice

Il lavoro aperto lo decide la capacità di review, non il numero di agenti

Verifica

Test derivati dai criteri di accettazione · due persone su ogni modifica

Rilascio

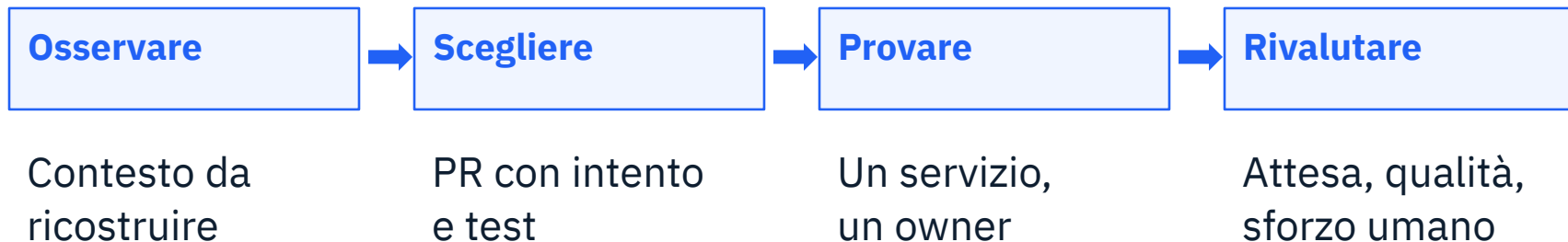
Plain old good engineering: Servizi autonomi - Deploy più frequenti

In 4 parole: Processi, Piattaforma, Automazione, AI



Come capiamo se stiamo migliorando davvero il lavoro, anziché rendere più veloce un singolo pezzo?

Un esperimento verificabile



Una data per decidere: mantenere, adattare o interrompere



Pull Request & Code Review

From a working change to a merged pull request. A smooth review process accelerates delivery and builds better software, together.

STAGE	1. PREPARE Get ready to submit	2. CREATE PR Open the pull request	3. AUTOMATED CHECKS CI, quality and security	4. REVIEW Get feedback	5. ITERATE Address feedback	6. APPROVE & MERGE Complete the change	7. FOLLOW UP Validate and learn
GOAL What I want to achieve	Ensure my changes are ready for review and easy to understand.	Create a clear, well-described pull request.	Pass automated checks (CI, tests, security, quality).	Get timely and valuable feedback from the right reviewers.	Address feedback efficiently and iterate.	Get approval and merge the change.	Confirm the change is deployed and working in production.
DEVELOPER ACTIONS What I do	<ul style="list-style-type: none"> Run local tests and linters Clean up commits Update documentation Check for breaking changes 	<ul style="list-style-type: none"> Create PR in GitHub Write clear title and description Link to issue (e.g. Jira) Add labels and reviewers Specify context and rationale 	<ul style="list-style-type: none"> Watch CI pipeline run Check test, build, security and quality results Investigate and fix failures 	<ul style="list-style-type: none"> Get code review from peers Discuss implementation details Clarify requirements Consider alternative solutions 	<ul style="list-style-type: none"> Make code changes Push new commits Respond to comments Re-request review if needed 	<ul style="list-style-type: none"> Ensure all approvals are in Merge (squash/rebase) Verify automatic deployment Close related issue 	<ul style="list-style-type: none"> Check deployment status Validate the feature in prod Monitor for issues Learn from feedback
TOUCHPOINTS & TOOLS Where I interact	<ul style="list-style-type: none"> IDE (e.g. VS Code) Local tests Linters / formatters Pre-commit hooks 	<ul style="list-style-type: none"> GitHub Jira (issue link) Contribution guidelines PR template 	<ul style="list-style-type: none"> CI/CD (GitHub Actions) Test results Security scans (SAST/DAST) Code quality (SonarQube, etc.) 	<ul style="list-style-type: none"> GitHub (reviews) Thread comments Design docs / ADRs Slack / Teams (async) 	<ul style="list-style-type: none"> GitHub (push & update) Comments Re-run CI Updated documentation 	<ul style="list-style-type: none"> GitHub (merge) Deployment pipeline Jira (transition) Release notifications 	<ul style="list-style-type: none"> Monitoring (e.g. Datadog) Logs User feedback (internal) Product metrics
KEY QUESTIONS What's on my mind	<ul style="list-style-type: none"> Do I have everything ready? Did I follow the guidelines? Is my code clear and focused? 	<ul style="list-style-type: none"> Is the description clear? Did I include enough context? Are the right people notified? 	<ul style="list-style-type: none"> Why did the build fail? What do I need to fix? Are these checks relevant? 	<ul style="list-style-type: none"> Who is the right reviewer? How long will it take? Is the feedback actionable? 	<ul style="list-style-type: none"> Do I understand the feedback? What's the best way to address it? When will it be reviewed again? 	<ul style="list-style-type: none"> Are all comments resolved? Can I merge safely? When will this be in production? 	<ul style="list-style-type: none"> Is the change working? Should I monitor anything? What can I learn for next time?
COMMON PAIN POINTS What can go wrong	<ul style="list-style-type: none"> Unclear requirements Missing tests Large or mixed changes Uncertainty about guidelines 	<ul style="list-style-type: none"> Vague PR description Missing context or links Wrong reviewers Inconsistent PR template usage 	<ul style="list-style-type: none"> Long CI queue times Flaky or unstable tests Hard to understand failures Environment inconsistencies 	<ul style="list-style-type: none"> Slow or missing reviews Flaky or subjective feedback Too many reviewers Bikeshedding / style nitpicks 	<ul style="list-style-type: none"> Multiple review cycles Conflicting feedback Outdated review comments Context switching 	<ul style="list-style-type: none"> Last-minute issues Merge conflicts Deployment failures Unclear release process 	<ul style="list-style-type: none"> Issues in production Missing observability Lack of feedback loop No recognition of contribution
DEVELOPER EXPERIENCE How it feels	 Focused	 Confident	 Frustrated	 Hopeful	 Challenging	 Satisfied	 Accomplished
OPPORTUNITIES (DEVEX) How we can help	<ul style="list-style-type: none"> Clear contribution guidelines Pre-commit checks Templates and examples IDE integrations 	<ul style="list-style-type: none"> Better PR templates Automated issue linking Smart reviewer suggestions Lightweight PR checklist 	<ul style="list-style-type: none"> Faster CI feedback More reliable and parallel tests Clearer error messages Cache and incremental builds 	<ul style="list-style-type: none"> Improve review SLAs Reviewer load balancing Review best practices Async review guidelines 	<ul style="list-style-type: none"> Better tooling for iterations Keep feedback actionable Resolve outdated comments Support for pair review 	<ul style="list-style-type: none"> Safer merge automation Clear deployment visibility Automatic issue closure Celebrate merged PRs 	<ul style="list-style-type: none"> Automatic post-deploy checks Easy access to logs/metrics Share impact (e.g. changelog) Capture learnings and insights
SUCCESS METRICS How we know it works	<ul style="list-style-type: none"> % of PRs with complete template % passing local checks Onboarding time to first PR 	<ul style="list-style-type: none"> % of PRs with issue link % with clear description Time to first review 	<ul style="list-style-type: none"> CI median duration Build success rate % flaky tests 	<ul style="list-style-type: none"> Time to first review Time to final review Review participation rate 	<ul style="list-style-type: none"> Average number of review cycles Time to address feedback % of PRs updated within 24h 	<ul style="list-style-type: none"> Time from approval to merge Deployment success rate Change failure rate 	<ul style="list-style-type: none"> % of changes without issues Time to detect issues Developer satisfaction (survey)

Perché la modifica aspetta?

Il disagio è un segnale. La causa richiede evidenze.

Ipotesi

Test instabili



Evidenza da raccogliere

Errori e riesecuzioni

Ambiente conteso



Richiesta, disponibilità, avvio

Decisione in attesa

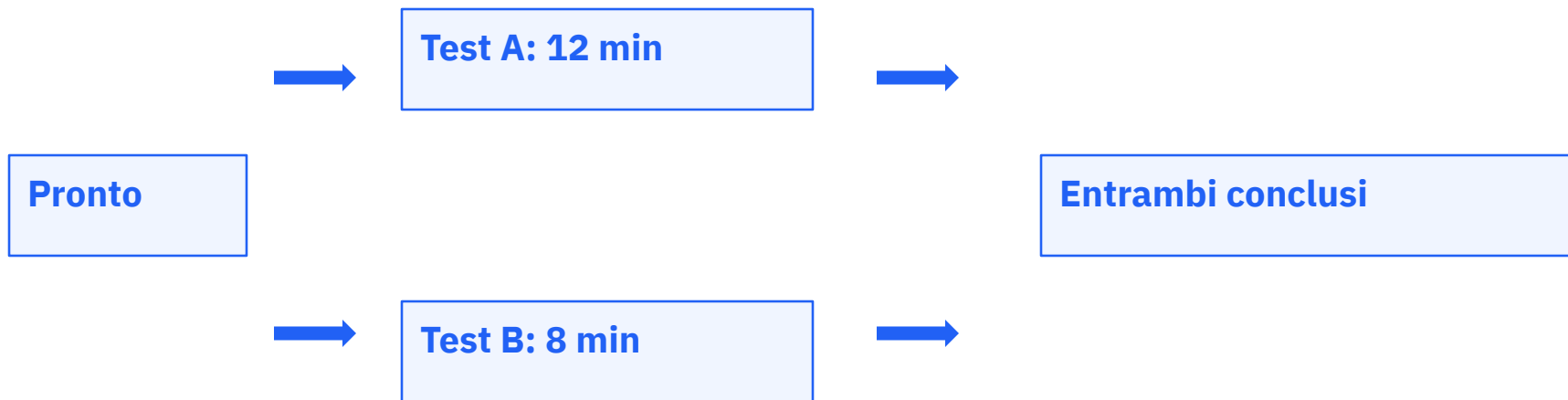


Evidenze pronte, decisione

CI lenta? Separiamo coda, esecuzione e riesecuzioni.



Il ramo che determina la fine



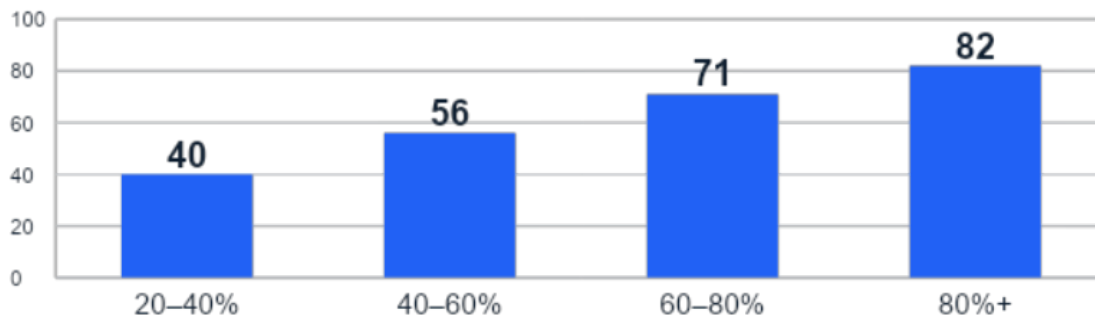
Test A scende a 6 min: il tempo dei test diventa 8 min

Esempio sintetico: avvio simultaneo, nessuna coda, entrambi obbligatori.



La coda prima della review

Attesa mediana prima della prima review, in minuti

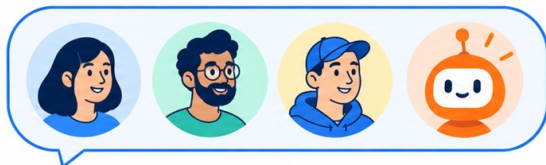


Quota di codice AI nelle organizzazioni

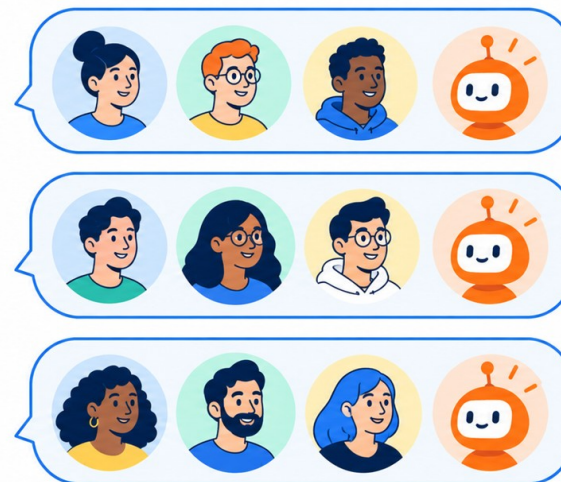
DX Q3 2026, luglio-settembre. Associazione tra gruppi, non causalità.

Claude Tag in Docebo

Pilota iniziale



Rollout in corso



Prime richieste: ambienti eseguibili e pipeline rapide

Osservazioni iniziali durante il rollout, senza risultati consolidati



Plugin Hub: skill condivise

Sviluppo e review

ai-sdlc, code-review

Conoscenza

codebase-docs, kb-skills

Task e difetti

jira-task-delivery, defect-resolution

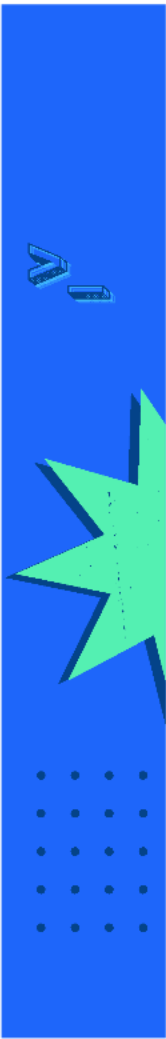
Processi

processes, skill e plugin di piattaforma

Plugin e skill prima dispersi, ora raccolti in un punto condiviso



Il contratto tra persone e agenti



E i junior?



docebo



AI Won't Fix Your Developer Experience.

But a great Developer Experience
will make AI much more valuable.



Continuiamo la conversazione

<https://www.linkedin.com/in/alexpagnoni/>

<https://www.linkedin.com/in/riccardo-galimberti/>

