1. Lemgo Techtalk

01.04.2026

Git as a Crime Scene

Präsentation von Oliver

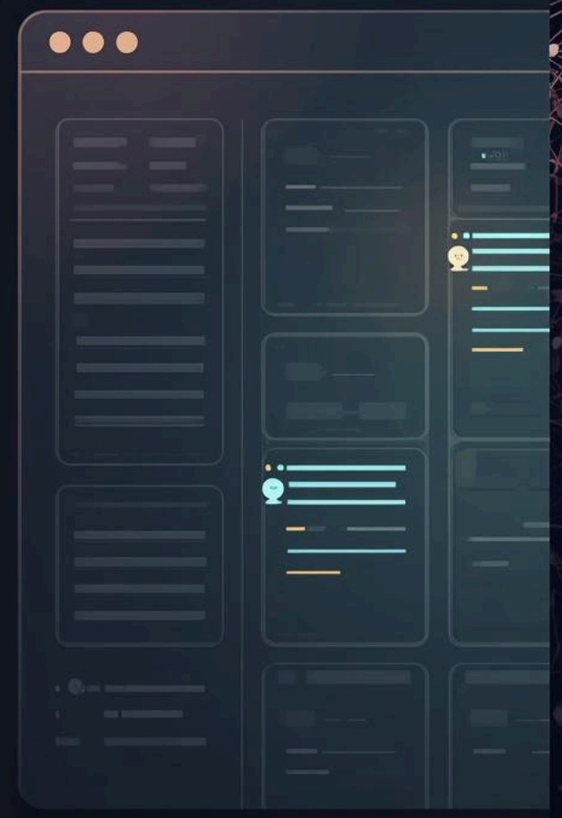
Meeting sagt:
Alles unter
Kontrolle

Git sagt:
Hotfix 03:17
auf main

Geplant vs. git log

Was sagt dein Repo?

Es ist entscheidend, den **echten Status** eines Projekts zu verstehen. Dein Git-Repo kann mehr Insights liefern als jedes Meeting.



GIT

Git als Verhaltensprotokoll verstehen

Code-Archiv

Jede Entscheidung, die wir im Code treffen, wird dokumentiert. Git ermöglicht es uns, die **Rationalität** hinter jeder Änderung nachzuvollziehen und unser Vorgehen zu hinterfragen.

Verhaltensprotokoll

Stresslevel können im Repository abgelesen werden. Häufige, hektische Commits deuten auf **Druck** im Team hin, während pausenlose Arbeit an großen Features auf ungelöste Probleme hinweist.

Was wir lesen können

Erkenntnisse aus unserem Git-Repo

Code Evolution

Die Entwicklung des Codes zeigt, wie sich Anforderungen geändert haben. Jede Zeile erzählt Geschichten über **Entscheidungen** und **Herausforderungen**, die das Team gemeistert hat, während es wächst.

Team Dynamics

Git offenbart die **Dynamik** im Team. Wie oft und wer commitet, zeigt nicht nur die Beteiligung, sondern auch, wie gut die Kommunikation innerhalb des Teams funktioniert.

Commit-Muster

Aktivitätshoch

Zu diesem Zeitpunkt zeigten die Entwickler eine **intensive** Aktivität im Repository.

Regelmäßige Beiträge

Über die Monate hinweg gab es **konstante** Beiträge, die das Team unterstützen.

Rückgang

In dieser Phase war die **Entwicklung** verlangsamt, was auf mögliche **Probleme** hinweist.

Autorstatistiken

Der **Bus Factor** ist entscheidend für das Verständnis deines Teams. Er zeigt, wie viele Personen für den Projektfortschritt unerlässlich sind. Ein niedriger Bus Factor bedeutet, dass das Wissen konzentriert ist, was Risiken birgt und die Teamdynamik beeinflusst.

git blame

Forensischer Ansatz zur Fehleranalyse

Forensische Analyse

Mit dem Befehl **git blame** kannst du nachvollziehen, wer für bestimmte Codezeilen verantwortlich ist. Fehlt die Verantwortlichkeit, hast du möglicherweise ein größeres Problem im Team oder Prozess.

Meta-Einblick

Git als Spiegel unserer Strukturen

Organisation sichtbar machen

Git offenbart nicht nur **Code**, sondern auch, wie Teams **arbeiten** und **kommunizieren**. Die Struktur und Dynamik der Zusammenarbeit wird durch Commit-Historien und Branch-Strategien sichtbar.

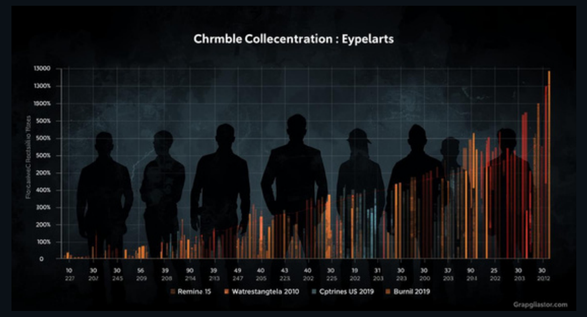
Nachverfolgbare Metriken für dein Projekt

Commit-Patterns

Sind anhand der commits die Änderungen Nachvollziehbar

Release-Chaos

Sind alle releases sauber gesteurt?



Merge-Complexity

Wie oft hängen große branches?

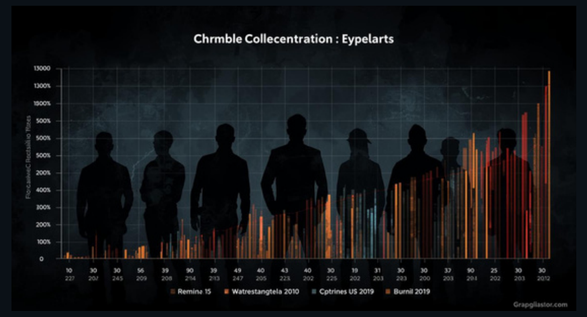
Nachverfolgbare Metriken für dein Projekt

Commit-Frequenz

Häufigkeit der Commits zeigt das Team-Engagement.

PR-Größe

Die Größe der PRs beeinflusst den Review-Prozess.



Eigentums-Konzentration

Wer ist für den Code verantwortlich und warum?

Wann wird hier gearbeitet?

```
git log --since="6 months ago"
--pretty=format:"%ad"
--date=format:"%H:00"
| sort | uniq -c | sort -nr | head
```

Commit Patterns / Stress sichtbar machen

Wer versteht dieses System wirklich?

```
git log --since="6 months ago" --pretty=format:"%an" | sort | uniq -c | sort -nr | head -5
```

Knowledge Islands

Wie sauber ist euer Release-Prozess?

Release Chaos sichtbar machen

```
git for-each-ref refs/tags \
--format='%(refname:short)|%(taggerdate)' \
| awk -F'|' '
function is_semver(tag) {
  return tag ~ /^v?[0-9]+\.[0-9]+\.[0-9]+$/
}
{
  total++

  # Lightweight vs Annotated
  if ($2 == "") lightweight++
  else annotated++

  # SemVer Check
  if (is_semver($1)) semver++
  else non_semver++
}
END {
  printf "Total tags: %d\n", total
  printf "Annotated tags: %d\n", annotated
  printf "Lightweight tags: %d\n", lightweight
  printf "SemVer compliant: %d\n", semver
  printf "Non-SemVer tags: %d\n", non_semver

  annotated_score = (annotated/total)*100
  semver_score = (semver/total)*100

  printf "\nAnnotated ratio: %.2f%%\n", annotated_score
  printf "SemVer ratio: %.2f%%\n", semver_score

  quality = (annotated_score * 0.6) + (semver_score * 0.4)
  printf "\nRelease Quality Score: %.2f / 100\n", quality
}'
```

Wie oft kracht es wirklich?

```
git rev-list --all --min-parents=2 --since="6 months
ago" \
| while read commit; do
    git show --shortstat --format="" "$commit"
  done \
| grep "files changed" \
| awk '{
    files += $1
    merges++
  }
  END {
    if (merges > 0) {
      printf "Total merges: %d\n", merges
      printf "Total files changed: %d\n", files
      printf "Avg files per merge: %.2f\n",
files/merges
    } else {
      print "No merge commits found in this period."
    }
  }'
```

Merge Complexity

Wo passiert die meiste Action?

```
git log --since="6 months ago"
--numstat --pretty="%H" \
| awk '
  { added[$3]+=$1;
deleted[$3]+=$2 }
  END {
    for (file in added)
      print
added[file]+deleted[file],
file
  }' \
| sort -nr \
| head -20
```

Hotspots / Risiko-Dateien

Die “besten” schlimmen Funde in Git

Secrets in Git
Man committet private Keys, Passwörter oder Tokens. Das ist nicht nur peinlich, sondern richtig gefährlich. Vor allem ChatGPT Keys

Kreative Branch-Namen
Klassiker wie fix, fix2, fix_final, fix_final_really. Oder dank Coding-Agents: my_feature_w7b84t5

Rebase Gone Wrong
Schnell ein rebase auf den aktuellen branch und “hups” den falschen branch zurückgesetzt.

Mindshift

Was sagt Git?

Wenn du den Status eines Projekts ermitteln möchtest, verlasse dich nicht nur auf Meetings. **Frage, was Git dir sagt.** Die Antwort liegt in den Commits und Historien.

Abschluss

Die Macht von Git in Projekten

Fragen?

Was sagt Git über eure Projekte?

Kontakt

Danke für deine Aufmerksamkeit!

Email: info@oliverfries.com
Website: oliverfries.com