Contao KONFERENZ 2024

# Effektive Code-Versionierung mit Feature-Branches

Best Practices und Strategien

Contao KONFERENZ 2024

# Jan Friebe

- 32 Jahre alt, aus NRW
- 9+ Jahre Erfahrung in der IT-Welt
- Team Lead & Lead Angular developer @ Eviden Germany GmbH
- Full Stack Entwickler, Schwerpunkt Frontend
- JS/TS // Vue // React // Angular // CSS // Design
- CrossFitter/Hobby Boxer/Runner/Daddy

jan-friebe.de friebe jan_friebe

# Fahrplan


1. Versionierung
2. Einführung in branching Strategien (Vor- und Nachteile)

**Tipps und best practises**

3. Branch Namenskonvention
4. Das approval System
5. PR Management
6. Commit Nachrichten
7. Nutzung von CI/CD
8. Tooltipps

Contao KONFERENZ 2024

# Versionierung

Wir versionieren Code um Änderungen nachverfolgen zu können, die Zusammenarbeit (im Team) zu erleichtern und stabile Entwicklungsprozesse mit klaren Release-Zyklen zu gewährleisten.

Contao KONFERENZ 2024

# Wie kann ich die Vorteile einer strukturierten Versionierung optimal für mein Projekt nutzen?

Mit der richtigen Strategie! Github Flow Git Flow

# Single-Branch-Strategie - Ein branch für alles!

*Versteht mich nicht falsch, es ist völlig in Ordnung, mit einem Zweig zu arbeiten, wenn du alleine arbeitest. Sobald du aber im Team arbeitest, reicht die Strategie meist nicht mehr aus*

# Nachteile der Single-Strategie

- ✕ Fehlende Isolation von Features
- ✕ Schwierige Fehlerbehebung/Lokalisierung
- ✕ Erhöhtes Konfliktrisiko
- ✕ Keine parallele Entwicklung
- ✕ Kein sauberer Release-Prozess

Wir büßen erheblich an Flexibilität ein!

Contao KONFERENZ 2024

# Multi-Branch-Strategie - Arbeiten mit Feature branches!

Contao 2024 KONFERENZ

# Vorteile der Multi-Strategie

- ✓ Isolation von Änderungen
- ✓ Parallele Entwicklung
- ✓ Saubere Versionskontrolle
- ✓ Konfliktvermeidung
- ✓ Bug fixing
- ✓ Kollaboration / Zusammenarbeit
- ✓ Bessere Rückverfolgbarkeit/Transparenz

# Multi-Branch Beispiele

- **Feature Branch**: Für neue Funktionen, z.B. `login-page`

Für jede Arbeit wird ein eigener, isolierte branch angelegt!

Contao KONFERENZ 2024

# Was für Änderungen enthält der branch ?

- `branch123`

Contao KONFERENZ 2024

# Branch Kategorisierung

features bugfix release hotfix experiment username docs config

Kategorisierung verhilt zu mehr Struktur und Effizienz im Entwicklungsprozess

Contao KONFERENZ 2024

# Branch prefix - easy

- **Feature Branch**: Für neue Funktionen, z.B. `feature/login-page`
- **Bugfix Branch**: Für die Behebung von Fehlern, z.B. `bugfix/fix-login-bug`
- **Hotfix Branch**: Dringende Korrekturen in der Produktion, z.B. `hotfix/security-patch`
- **Refactoring Branch**: Um Code zu verbessern, z.B. `refactor/user-authentication`
- **Documentation Branch**: Für Änderungen an der Dokumentation, z.B. `docs/update-api-docs`
- **Release Branch**: Um eine Version vorzubereiten, z.B. `release/v1.0.0`

Contao KONFERENZ 2024

# Das approval System

Ein Verfahren bei dem Code-Änderungen über Pull-Requests zur Überprüfung vorgelegt werden, bevor sie in den Hauptbranch integriert werden.

Contao KONFERENZ 2024

# Vorteile des approval Systems

- ✓ Sicherstellung von Code-Qualität
- ✓ Früherkennung von Fehlern und Problemen
- ✓ Förderung von Tranzparenz und Zusammenarbeit im Team
- ✓ Gewährleistung und Einhaltung von Standards und Richtlinien

Contao KONFERENZ 2024

# Tipps für Pull Requests

1. Klare Beschreibungen und Kontextinformationen
2. Kleine, fokussierte Änderungen
3. Automatisierte Tests und Linting
4. Konstruktives Feedback geben und annehmen

Contao KONFERENZ 2024

# Semantische Commits

Commit Nachrichten die nach einem bestimmten Muster verfasst werden um den Zweck der Code-Änderung eindeutig zu kennzeichnen.

Contao KONFERENZ 2024

# Beispiel

```
feat(user-authentication): Add OAuth2 login functionality
Implement OAuth2 authentication using Google Sign-In Api
- Configure OAth2 client Id and secret
- Add login button to the user
```

type = feat

scope = user-authentication

short summary = Add OAuth2 login func

Longer description = Detailed description & bullet points

feat | fix | docs | style | refactor | test (adding or modifying test) | core (other changes related to build e.g. maintenance tasks

# CI/CD (kontinuierliche Integration)

- Automatisierte Tests in jedem Branch
- Kontinuierliche Bereitstellung (CD)
- Pull-Requests und Quality Gates
- Schnelle Bereitstellungszyklen

# Tools und Unterstützungen

- **Husky**: Pre-Commit-Hooks, z.B. für einheitliche Commit-Nachrichten
- **Lint-Staged**: Lintet nur geänderte Dateien vor einem Commit
- **CI/CD-Tools** (z.B. Jenkins, GitLab CI, GitHub Actions): Automatisiertes Testen und Build-Prozess

- **Semantic Release**: Automatische Versionierung und Changelog-Erstellung basierend auf Commits
- **Prettier/ESLint/biome/**: Einheitliche Code-Formatierung und Qualitätsprüfung
- **Dependabot**: Automatisierte Updates von Abhängigkeiten in Branches
- **Branch Naming Conventions Tools** (z.B. Commitizen): Erzwingt konsistente Branch-Namen und Commit-Nachrichten
- **Review Tools** (z.B. SonarQube): Automatische Code-Qualitätsanalyse

Danke für eure Aufmerksamkeit. Fragen? ❤️