PLOS
A catalyst for better

Rapid Prototyping für digitale Verlagsprodukte mit Vibe Coding und KI-Unterstützung

What could possibly go wrong? ;-)

PLOS
Christian Kohl
Director Technology & Engineering

CrossMediaForum, München, 3.7.2025

Title

1. Was ist Vibe Coding?
2. Voraussetzungen
3. Wie funktioniert das?
4. Was bringt das?
5. Risiken und Nebenwirkungen
6. “Best practice” / Tipps & Tricks
7. Was ist sonst noch wichtig?

Was ist Vibe Coding?

- Konversationsbasierte Softwareentwicklung
- “stream of consciousness”, intuitiv
- Interaktiv, iterativ, inkrementell
- Für Laien / nicht-Techniker*innen

Quelle: https://x.com/karpathy/status/1886192184808149383?lang=en

Wer hat’s erfunden?

Voraussetzungen

- Iterative & Interaktive Kollaboration mit der KI
- Mit unerwarteten Ergebnissen umgehen, Ambiguitätstoleranz
- Ergebnisse bewerten & formen/verfeinern (“Evolution”) anstatt vorhersehbare, fertige Liefergegenstände erhalten
- Zugang zu einem der zahlreichen Tools (Replit, Cursor, Lovable, Claude Code, …)

+ Zeit, Neugier, Geduld ;-)

Wie funktioniert das?

- Software diskursiv entwickeln lassen

Plan — Build

I want a workout timer app for android with minimalistic design. Workouts sollen in Sätze unterteilt werden können, die App soll offline funktionieren und der Timer soll 3 Sekunden vor Ablauf des Satzes einen Countdown starten.

CK

Just now

I understand you want to build a workout timer app for Android with specific requirements. Let me analyze what you're trying to build:

Your Objective:
You want to create a minimalistic Android workout timer app that:

- Allows workouts to be divided into sets (Sätze)
- Works completely offline
- Features a countdown that starts 3 seconds before each set ends
- Has a clean, minimalistic design

However, I need to let you know that our current development environment is optimized for web applications rather than native Android development. Android app development with native tools like Android Studio, Kotlin, or Java isn't available in our current stack options.

Alternative Solution:
I can propose building a **web-based workout timer** that would work excellently on mobile devices through the browser. This would give you:

- A mobile-responsive design that works great on Android phones
- Offline functionality through web technologies
- All your requested timer features
- Clean, minimalistic interface
- The ability to install it as a Progressive Web App (PWA) on Android

Would you like me to create a plan for a mobile-optimized web application that delivers the same workout timer functionality you're looking for? This would be accessible on any Android device through the browser and could even be added to the home screen for a native app-like experience.

Message Agent...

Android device through the browser and could even be added to the home screen for a native app-like experience.

Ja CK

3 minutes ago

Perfect! I'll create a plan for a mobile-optimized web-based workout timer app that delivers all the functionality you're looking for.

Plan Latest 17 minutes ago

FitnessFlow

Tech stack Modern web app

Initial version

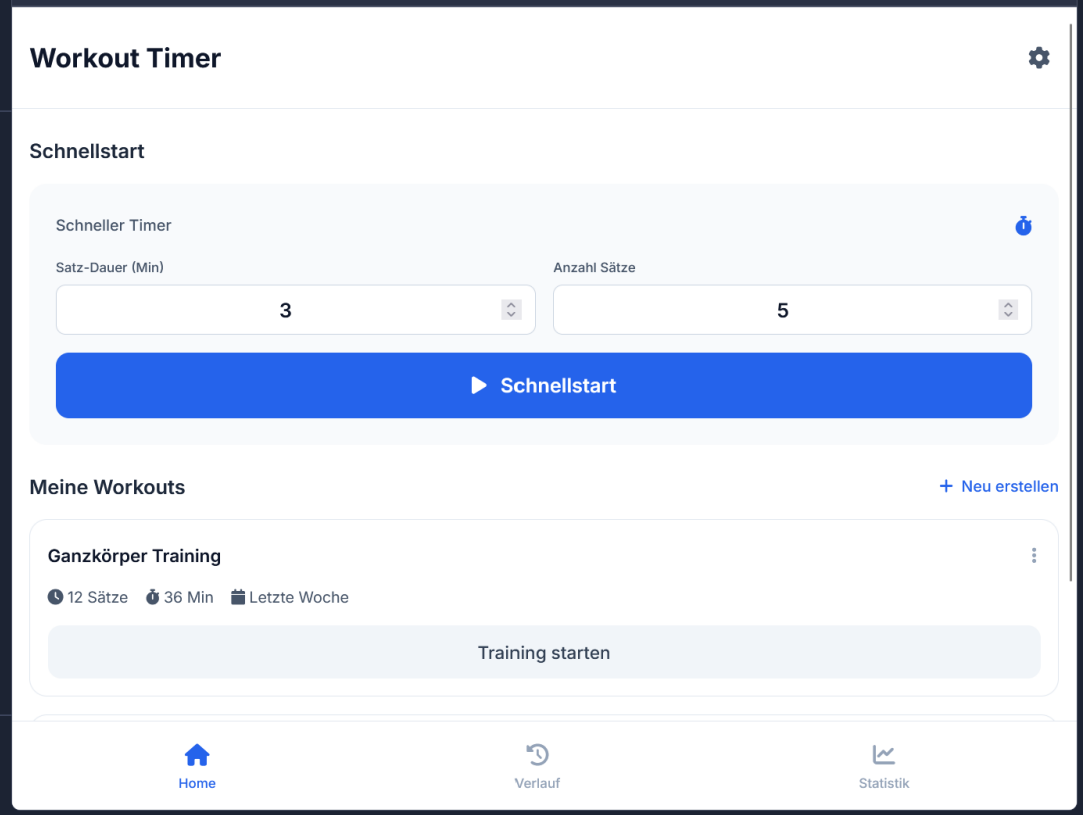
A minimalistic mobile-optimized workout timer web app with set-based workouts and offline functionality

- Create and manage workout routines with multiple sets
- Set-based workout structure (Sätze) with customizable durations
- Offline functionality - works without internet connection
- Timer with 3-second countdown warning before set completion
- Audio/visual alerts for set transitions
- Pause, resume, and stop workout controls
- Progress tracking during workouts
- Simple workout history

Later

Agent can work on these tasks after the initial version

Visual preview Agent will use this as a design reference

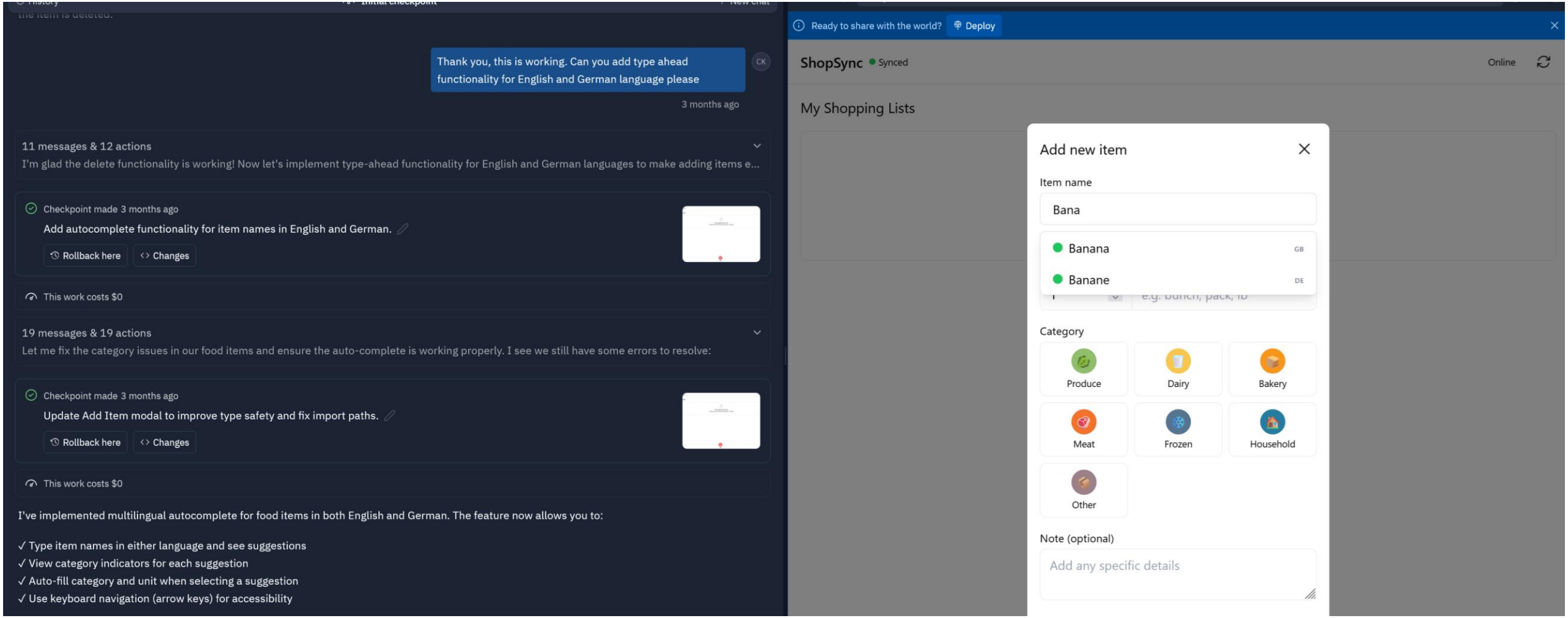


Review the plan & visual preview. If everything looks good, click approve to start building. You can also make revisions by chatting with me.

Keep chatting | Approve plan & start Ctrl ↵

Waiting for your response

Schrittweise verbessern / ausbauen



Was bringt das?

- Versetzt es nicht-Programmierer*innen in die Lage, den Design- und Prototyping-Zyklus zu beschleunigen
- Interaktives, Inkrementelles Design und Prototyping → Frühe und ständige Verprobung möglich, kurze Feedbackzyklen
- Ersetzt nicht die Entwicklungsabteilung oder IT – kann diese aber unterstützen und in bestimmten Phasen entlasten

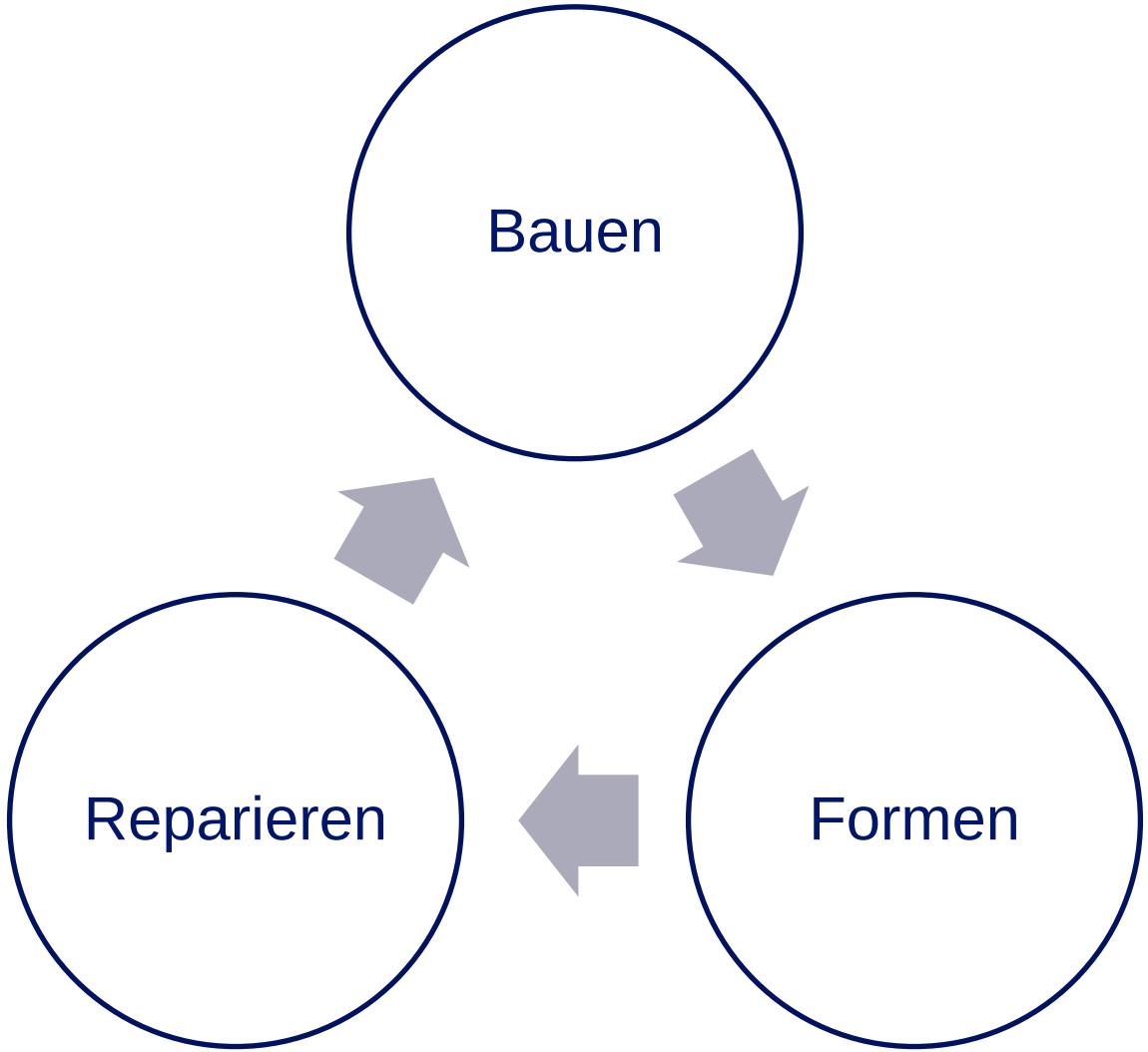
Drei grobe Nutzergruppen

1) Einsteiger*innen / Nicht-Techniker*innen
2) Entwicklungsnahe Tätigkeiten, bspw. Produktmanagement
3) Software Entwickler*innen

Risiken und Nebenwirkungen

- Explosion der Schatten-IT
- Sicherheitslücken, Complianceverstöße, Datenschutzverstöße, Performanceprobleme
- Langfristige Wartbarkeit, Weiterentwicklung
- Erhöhte Integrationskosten
- Verdummung

“Best practice” / Tipps & Tricks



“Bauphase” / Start

1. Kategorie und Domäne festlegen (“Kalendar”, “Taskmanager”, etc. | “Gesundheit”, “Bildung”, etc.)
2. 2-3 Schlüsselfeatures definieren
3. Grobe Vorgabe für Style / Ästhetik des UIs machen (“minimalistisch”, “verspielt”, farbenfroh”, “businesslike”, …)
4. Zuviel Detail vermeiden (Kontextfenster klein halten): Klein anfangen und inkrementell ausbauen
5. Komplizierte Features wie Authentifizierung, Datenbankanbindung, … erst am Schluss hinzufügen (sofern benötigt)

“Formen”

1. Mit kleinen Änderungen oder Verbesserungen starten
2. Immer nur eine Sache verändern (kleine Häppchen)
3. Schwierige Features erst zum Schluss
4. Existierende Services / APIs für Features nutzen, wo möglich (bspw. Emailversand via sendgrid o.ä. statt eigenem Emailserver aufsetzen)

“Reparieren”

1. Screenshots mit Annotationen sind gut für Feedback
2. Fehlermeldungen teilen und Erwartung klar formulieren
3. Den Agenten auffordern, die Logs während des Testens zu prüfen und Debug-Infos zu liefern
4. Den Agenten auffordern, Test-Tools zu bauen
5. Nicht zu tief reinsteigern / verwursten - besser wegwerfen und neu starten!

Allgemein

1. Von Zeit zu Zeit redundanten Code löschen lassen
2. Vorsicht bei Themen rund um Datenbanken: Nur weil der Agent sagt, er hätte eine DB verwendet, stimmt das noch lange nicht …
3. Nach größeren Änderungen die komplette App testen (Regressionen)
4. Gründlich testen, insb. auf fehlende Funktionalität (bspw. UI-Feature ohne passendes Backend)

Was ist sonst noch wichtig?

Organisation, Ethik

- Die Grenzen zwischen Produktmanagement, Design/UX und (Frontend-)Entwicklung müssen neu definiert werden, Rollenverständnis verändert sich.
- Neue Formen der Kollaboration
- Energieverbrauch und grundsätzliche Fragen zur Genese der Modelle

Sicherheit, Skalierbarkeit, Wartbarkeit, Compliance

- Debugging
- Suboptimaler Code
- Abhängigkeiten & Integrationen
- Verantwortung
- Geistiges Eigentum

Und die Softwareentwicklung?

- IDEs verändern sich
- Testen mit/durch Agents wird zunehmen
- Externe API Services (bspw. Paragon) gewinnen an Bedeutung
- Verschiebung von Rollen und Veränderung der althergebrachten Kollaborationsprozesse

Fazit

- Ernsthaftes Prototyping!
- Ideen früh verproben (mit echten Nutzer*innen) und dann inkrementell verbessern
- Goodbye „Big Bang“
- Automatisiertes Testen von Anfang an

A catalyst for better

Bonus Links zum Thema:

Ein lustiges Video: https://youtu.be/6x3weFV9YI4

Ein schönes Kunstprojekt von Rick Rubin: https://www.thewayofcode.com/