
KI-gestützte Legacy-Code-Portierung
MODERNISIERUNG VON SOFTWARE
DURCH KÜNSTLICHE INTELLIGENZ

1

Einführung in Legacy Code

Was ist Legacy und warum sollte alter Code portiert werden?

Einführung in Legacy-Code

Definition von Legacy-Code
Legacy-Code beschreibt Software, die veraltet ist, aber weiterhin in Betrieb bleibt und oft nicht mehr gewartet wird.

Herausforderungen bei der Wartung
Die Wartung von Legacy-Code kann schwierig sein, da die ursprünglichen Entwickler möglicherweise nicht mehr verfügbar sind und die Dokumentation fehlt.

Modernisierung von Systemen
Unternehmen stehen vor der Herausforderung, Legacy-Systeme zu modernisieren, ohne den laufenden Betrieb zu beeinträchtigen.

Warum ist eine Modernisierung aus technischer Sicht erforderlich?

Wettbewerbsfähigkeit erhalten
Die Modernisierung von Systemen ist entscheidend, um auf dem aktuellen Markt wettbewerbsfähig zu bleiben und neue Chancen zu nutzen.

Effizienzsteigerung
Neue Technologien ermöglichen es Unternehmen, ihre Prozesse zu optimieren und effizienter zu arbeiten, was Zeit und Ressourcen spart.

Skalierbarkeit und Kompatibilität
Durch die Modernisierung können Unternehmen bessere Benutzererlebnisse schaffen, was zu höherer Kundenzufriedenheit und Loyalität führt.

Warum ist eine Modernisierung aus betriebswirtschaftlicher Sicht erforderlich?

Kostenreduktion

Die Implementierung neuer Technologien trägt zur Senkung der Betriebskosten bei, was es Unternehmen ermöglicht, ihre Rentabilität zu steigern, da die Pflege veralteter Software oft mit hohen Ausgaben verbunden ist.

Sicherheitsrisiken und Performance-Probleme

Alte Systeme weisen Sicherheitslücken auf, die Cyberangriffe begünstigen. Zudem können sie Leistungsprobleme verursachen, wenn sie den Anforderungen neuer Anwendungen nicht mehr entsprechen.

Skalierbarkeit und Kompatibilität

Moderne Systeme ermöglichen eine bessere Anpassung an steigende Anforderungen und sind kompatibler mit neuen Technologien und Standards.

2

Prozessübersicht

Grundlegender Prozess

Code Bewertung und Dokumentation

Code Struktur
Die Analyse der Code-Struktur ist entscheidend für die Identifizierung von Schwächen und Verbesserungsmöglichkeiten innerhalb des bestehenden Codes.

Funktionalität verstehen
Ein Verständnis der Funktionalität stellt sicher, dass der Code wie erwartet arbeitet und die gewünschten Ergebnisse liefert, auch nach einer Portierung.

Dokumentation erstellen
Eine klare Dokumentation der alten Architektur hilft dabei den Ursprünglichen Code zu verstehen, Wissen zu bewahren und erleichtert zukünftige Anpassungen und Portierungen.

Ansätze für die Portierung

Big Bang-Portierung
Die Big Bang-Portierung erfolgt durch die sofortige Umstellung auf ein neues System, was schnelle Ergebnisse erzielt.
Jedoch je komplexer die Architektur, desto mehr Risiko entsteht bei der Portierung.

Inkrementelle Portierung
Bei der inkrementellen Portierung erfolgt die Umstellung schrittweise, was eine bessere Kontrolle und weniger Risiken ermöglicht, jedoch mehr Zeit in Anspruch nehmen kann. Entwickler können jeden Schritt nachvollziehen.

Richtige Wahl?

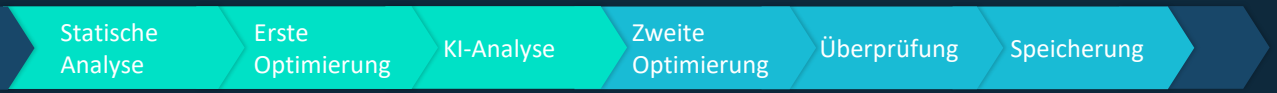
Zurzeit ist der Inkrementelle Ansatz notwendig. In der Zukunft, können durch fortgeschrittene KI-Modelle auch komplexere Architekturen auf einmal bearbeitet werden.

Zusammenspiel von Analyse-Tools, KI und Entwicklern

Analyse-Tools
Nutzen von statischen Code-Analysetools (z. B. SonarQube, Pylint), um strukturelle Schwächen und potenzielle Fehler frühzeitig zu erkennen und sofort zu beheben.

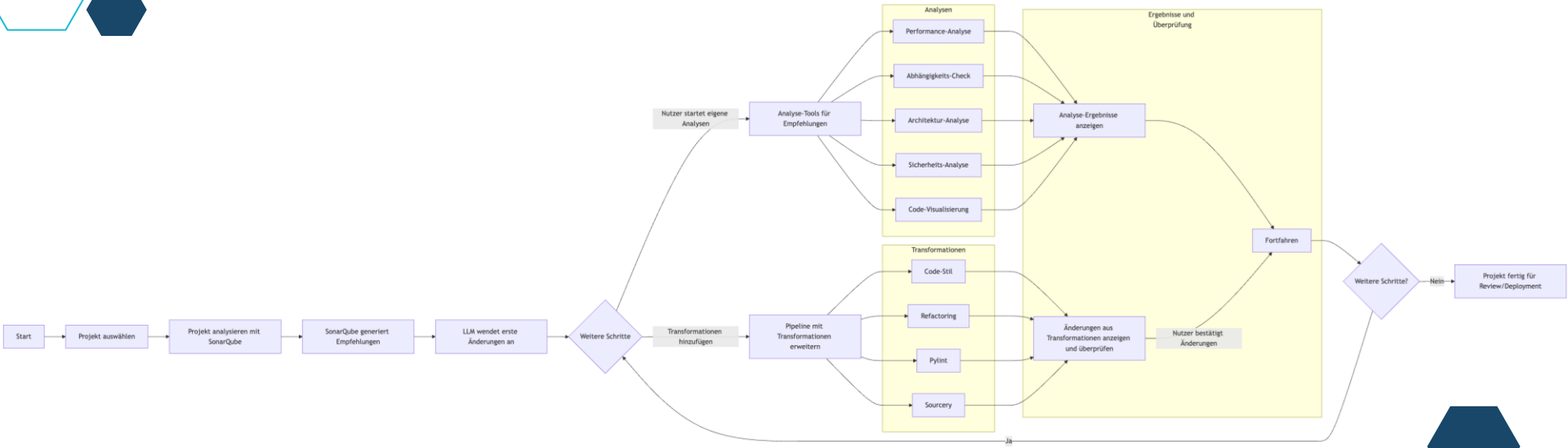
KI-gestützte Methoden
Automatisierte Vorschläge für Optimierungen, Code-Refactoring und Mustererkennung, um die Effizienz zu steigern und komplexere Probleme zu optimieren.

Menschliche Expertise
Entwickler evaluieren und verbessern die Resultate der KI- und Toolanalyse, um sicherzustellen, dass alle geschäfts- und anwendungskritischen Anforderungen erfüllt werden.



3 Technik und Implementierung

Allgemeiner Workflow



Analyse mit SonarQube

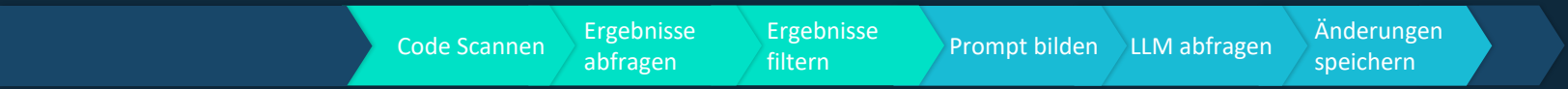
Analysebereich

- Codequalität
- Sicherheitslücken
- Codeabdeckung
- Technische Schulden
- Code Duplikate
- Abhängigkeiten
- Regelkonformität

Analyse mit SonarQube

Ablauf

- Scanner prüft aktuellen Code auf feste Regeln
- Ergebnisse über WebAPI abgefragt und gefiltert
- Gefilterte Ergebnisse werden als Prompt ans LLM weitergegeben



Analyse mit Pylint

Analysebereich

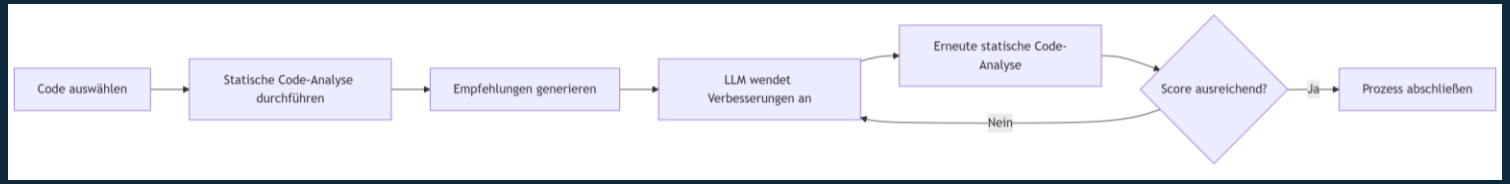
- Code-Stil
- Syntaxfehler
- Refactoring-Hinweise
- Fehlende Imports
- Undefinierte Variablen
- Komplexitätsbewertung

Analyse mit Sourcery

Analysebereich

- Automatisches Refactoring
- Optimierung von Schleifen und Bedingungen
- Entfernung redundanter Code-Strukturen
- Verbesserung der Lesbarkeit
- Vorschläge für Pythonic Code
- Konsolidierung ähnlicher Funktionen
- Verbesserung der Performance

Nutzung von statischer Code-Analyse und KI-Transformation



Benutze Tools für Statische-Code Analyse

SonarQube

Identifiziert Code-Schwächen, Sicherheitslücken und ermöglicht eine umfassende Bewertung der Codequalität. Unterstützt bei der Regelkonformität und technischen Schulden.

Pylint

Analysiert Python-Code auf Fehler, Code-Stil und Best Practices.
Liefert detailliertes Feedback zur Verbesserung der Lesbarkeit und Wartbarkeit.

Sourcery

Automatisiert Code-Verbesserungen mit KI-gestützten Vorschlägen für Refactoring und Performance-Optimierung. Ermöglicht die direkte Verbesserung des Codes durch intelligente Vorschläge.

Wahl des richtigen Modells

Qualitätsmerkmale

- **Qualität** der Ergebnisse für erstklassigen Code
- **Geschwindigkeit** für minimale Wartezeiten
- **Kosten** für günstige Skalierbarkeit und Zugänglichkeit

Siehe Kapitel: Weitere Ressourcen

Multi-Chain-Comparison

Beispiel

Versuch / Temperatur	Start	Erster	Zweiter
0.3	4.6	7.8	9.6
0.6	4.6	7.8	9.3
1.0	4.6	7.3	8.6

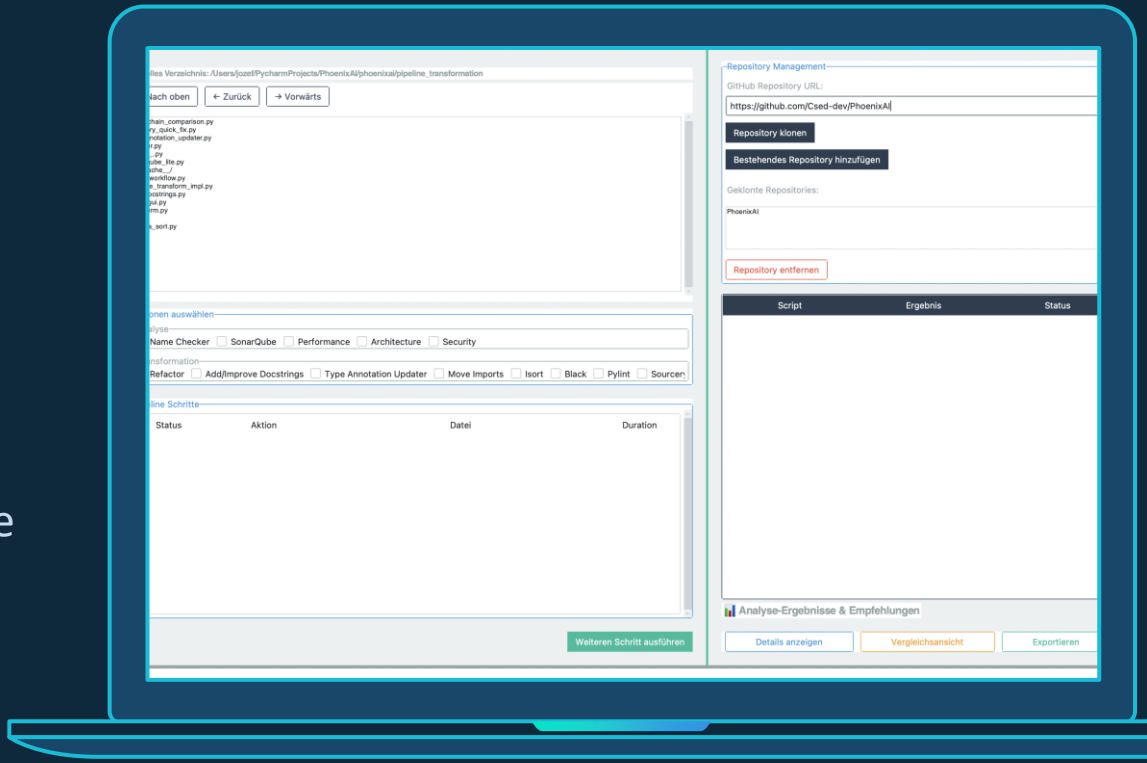
4

Anwendungsbeispiel

Wie benutzt man die Software?

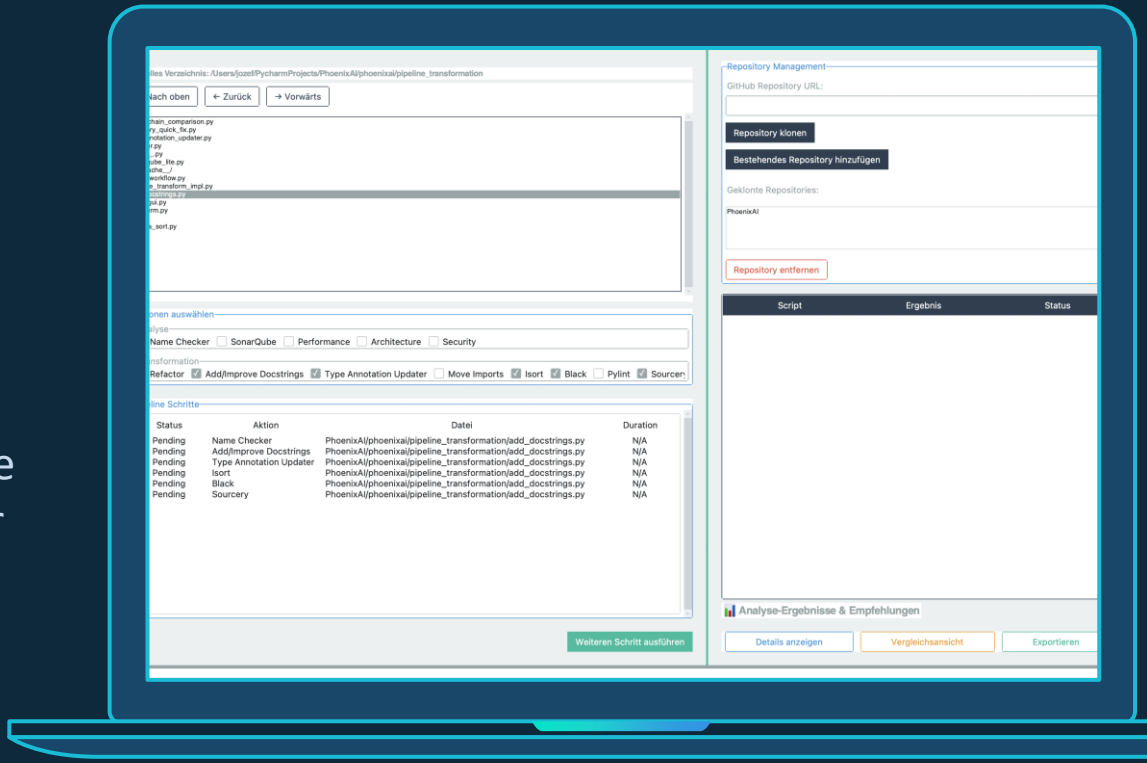
Anwendungsbespiel 1

1. Projekt aus GitHub klonen, oder eine lokale Kopie einbinden.
2. Zur gewünschten Datei navigieren



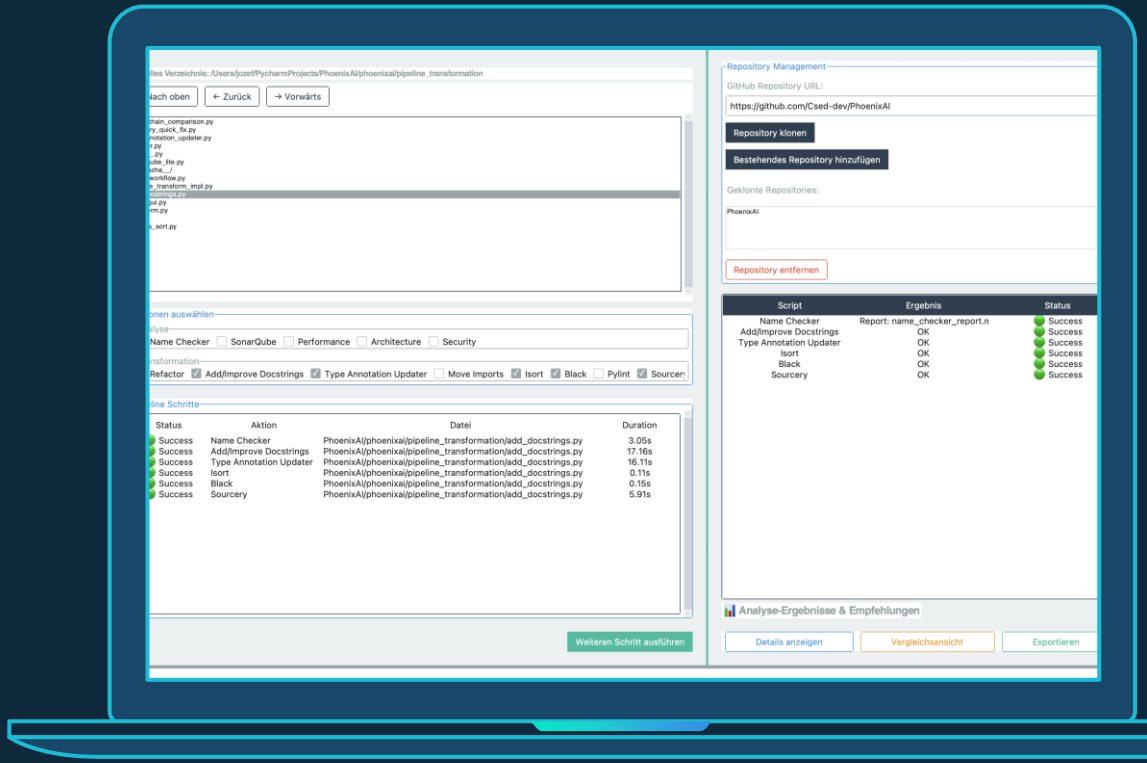
Anwendungsbespiel 2

1. Analyse- oder Transformationsschritte auswählen, die man für eine Datei ausführen möchte.
2. Dateien auswählen



Anwendungsbespiel 3

1. Einzelne Schritte ausführen.
2. Nach jedem Schritt das Ergebnis überprüfen



5 Potential

Was ist noch möglich?

- Integration in DevOps und CI / CD-Workflows
- Big Bang-Portierung
- Kleinere Architekturverbesserungen
- Aufteilen einzelner Klassen in größere Klassenstrukturen
- Automatische Dokumentation des überarbeiteten Codes

Kombination

Durch die Verbindung von bewährten Analyse-Tools, KI und Entwickler, kann ein optimales Ergebnis erzielt werden

6 Langfristige Strategien

63 % planen Modernisierung

Ein Großteil der Unternehmen will veraltete Systeme innerhalb der nächsten drei Jahre ersetzen.

>50 % geschäftskritische Altsysteme

Über die Hälfte der Legacy-Systeme ist essenziell für Unternehmensprozesse – ein hohes Risiko bei Ausfällen.

59 % höherer Wartungsaufwand

Altsysteme treiben Kosten durch ineffiziente Wartung in die Höhe, besonders bei kleinen Unternehmen.

https://www.thinkwisesoftware.com/de/blog/neue-studie-legacy-modernisierung-von-computerwoche
https://blog.workday.com/de-de/was-ist-legacy-it-und-warum-sie-jetzt-ihr-erp-altsystem-uberdenken-sollten.html
https://www.cio.de/a/it-steinzeit-in-deutschen-firmen%2C3553612

Business Model Canvas

Wichtige Partner

- Open-Source-Community für Beiträge, Feedback und Weiterentwicklung
- Anbieter von Open-Source-Tools und –Plattformen
- Hochschulen und Forschungseinrichtungen für Unterstützung und Promotion

Wichtige Aktivitäten

- Entwicklung und Pflege der Portierungspipeline mit Open-Source-Tools
- Bereitstellung von Tutorials und Dokumentation für einfache Nutzung
- Aufbau und Förderung einer aktiven Community für Verbesserungen

Schlüsselressourcen

- Open-Source-Tools wie SonarQube, Sourcery und Gemini-Flash
- GitHub für den Quellcode
- Community-Mitglieder, die aktiv zur Entwicklung beitragen
- Dokumentation und Beispiele, die den Einstieg erleichtern

Wertversprechen

- Effiziente und zugängliche Portierung von Legacy-Code mit Open-Source-Methoden
- Automatische Anpassung von Code an Google Code Guidelines
- Kostenlos nutzbare und anpassbare Software
- Förderung von Kollaboration und Wissensaustausch in der Entwicklergemeinschaft
- Transparenz und Mitgestaltungsmöglichkeiten für alle Beteiligten.

Kundenbeziehungen

- Enge Zusammenarbeit mit der Community durch Foren, Chats und Pull-Requests
- Bereitstellung von umfassender Dokumentation und Anleitungen
- Organisation von Hackathons und Community-Treffen zur Förderung der Zusammenarbeit.

Kanäle

- Veröffentlichung auf Plattformen wie GitHub
- Verbreitung über Open-Source-Foren und soziale Netzwerke
- Präsentationen bei Konferenzen, Hackathons und in Hochschulseminaren

Kundensegmente

- Entwickler, die Legacy-Code modernisieren wollen
- Unternehmen, die Open-Source-Lösungen für ihre IT-Modernisierung suchen
- Hochschulen und Studierende im Bereich Informatik und Softwareentwicklung, die an Open-Source-Projekten mitwirken wollen.
- Kleine Unternehmen, die Ihren Code standardisieren möchten

Kostenstruktur

- Hosting-Kosten für Plattformen und Dokumentation
- Zeitaufwand für Entwicklung, Pflege und Community-Management
- Organisation von Events wie Hackathons

Einnahmequellen

- Spenden durch Plattformen wie OpenCollective oder Patreon
- Förderung durch Hochschulen, Organisationen oder Stiftungen
- Einnahmen durch optionale Schulungen oder Beratungen
- Einnahmen durch die Entwicklung maßgeschneiderter Lösungen für Unternehmen

SWOT-Analyse

Stärken

Effiziente Kombination von KI-Algorithmen und Analysetools für skalierbare Portierungen. Modular, flexibel und an moderne Code-Standards angepasst.

S

W

Schwächen

Komplexität der Pipeline-Entwicklung. Abhängigkeit von KI-Modellen und anderen Analyse-Tools. Mangelnde Testabdeckung.

O

T

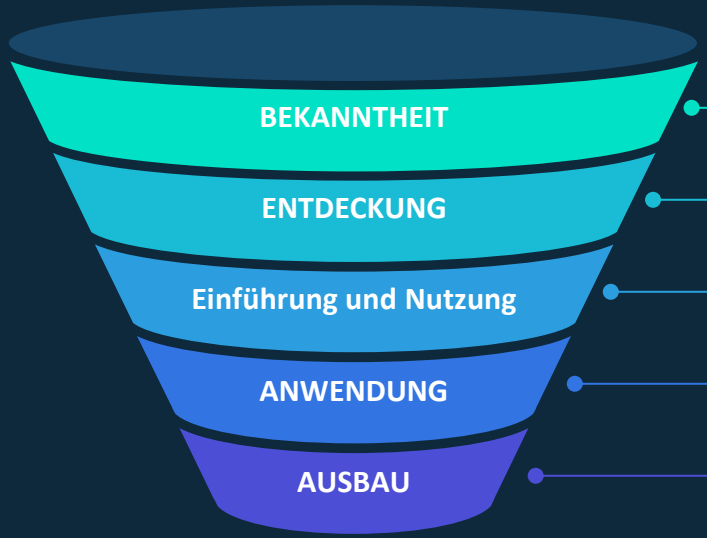
Wachsende Nachfrage nach IT-Modernisierung und Erweiterung auf Bereiche wie Code-Optimierung und Sicherheitsanalysen

Chancen

Verlust der Funktionalität. Inkompatibilität mit neuen Umgebungen. Risiko von neuen Sicherheitslücken.

Gefahren

Funnel



Die Zielgruppe wird durch Vorträge, Präsentationen und Online-Plattformen wie GitHub auf das Projekt aufmerksam gemacht.

Potenzielle Nutzer lernen das Projekt und seine Funktionen durch Demos, Beschreibungen und erste Einblicke kennen.

Einfache Tutorials, klare Dokumentation und eine einladende Community ermöglichen es, das Projekt schnell auszuprobieren und in die eigene Arbeit zu integrieren.

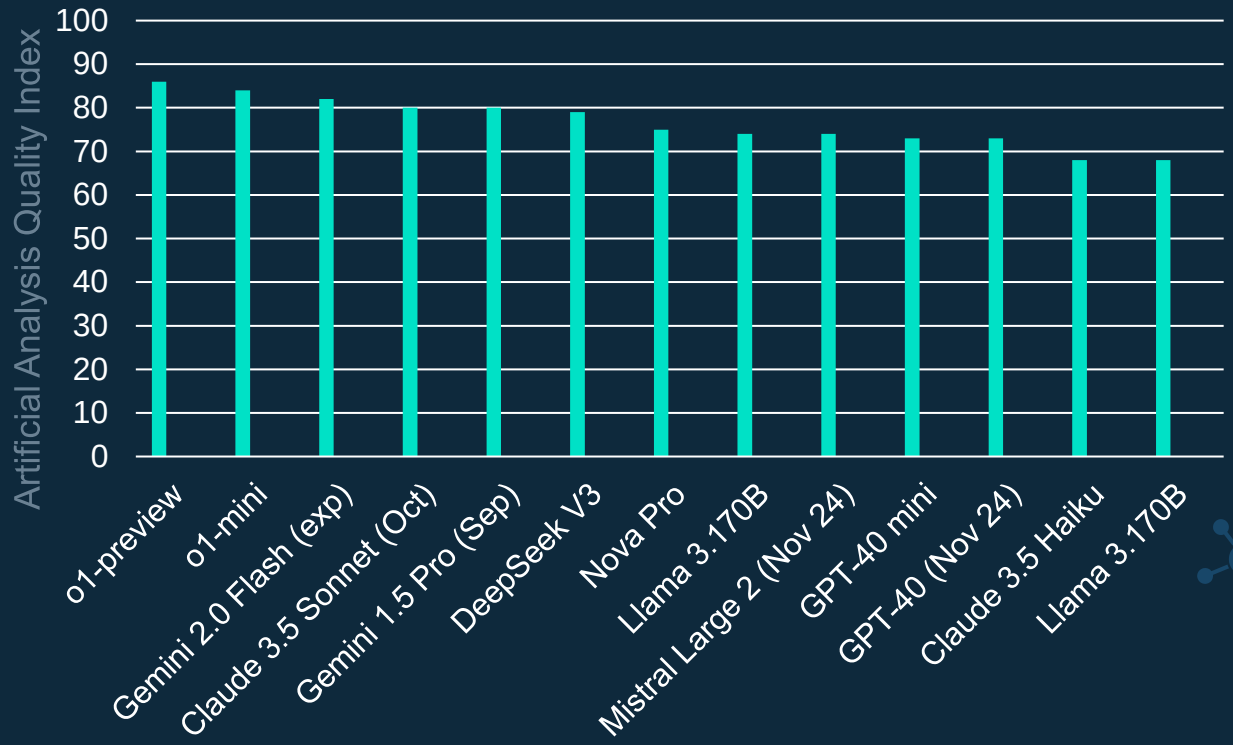
Nutzer setzen das Projekt aktiv für die Analyse und Portierung von Legacy-Code ein. Der Open-Source-Charakter ermöglicht Anpassungen an individuelle Anforderungen.

Das Projekt wird ständig durch Feedback aus der Community, und gemeinsame Entwicklungen optimiert.

8

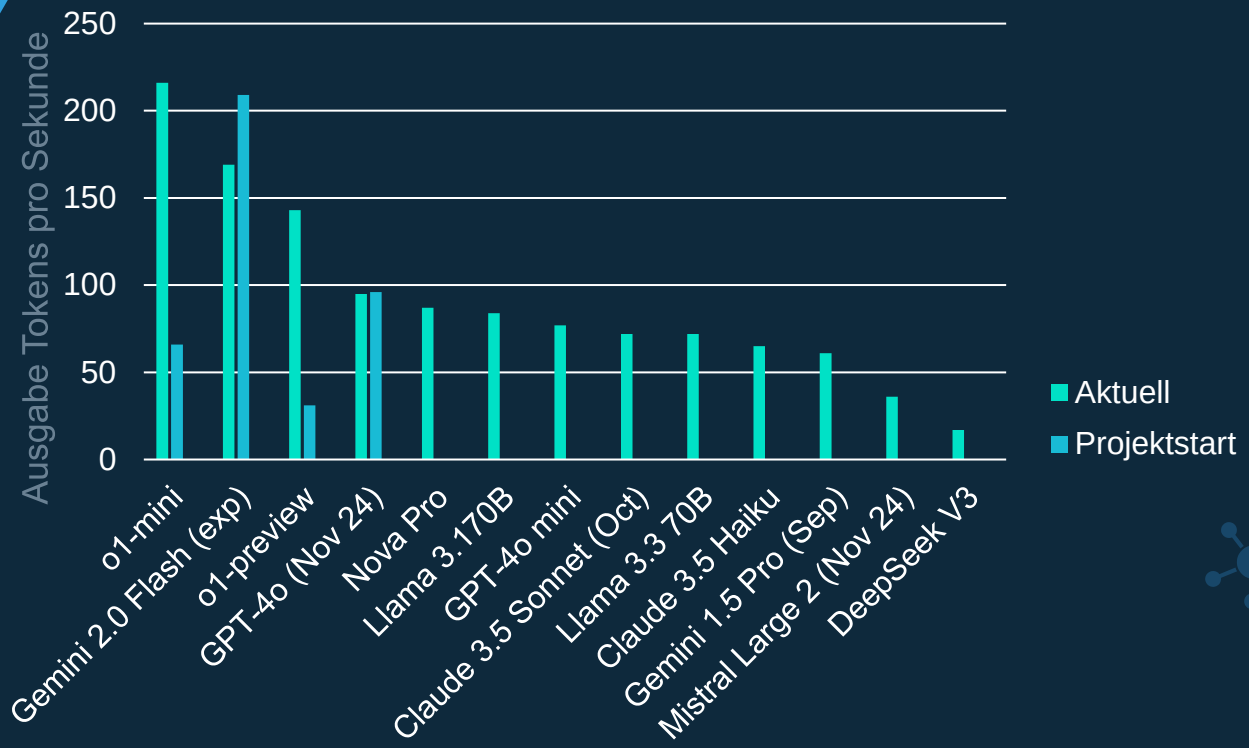
Weitere Ressourcen

Qualität von LLMs



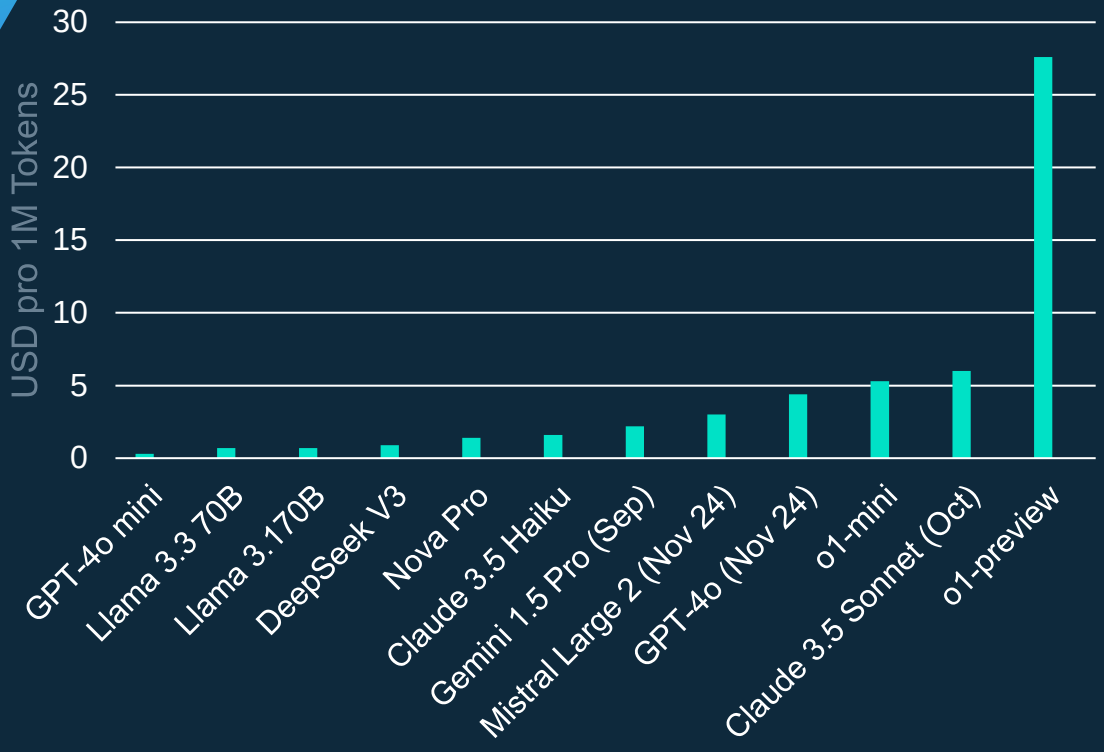
https://artificialanalysis.ai/models

Geschwindigkeit von LLMs



https://artificialanalysis.ai/models

Preise von LLMs



https://artificialanalysis.ai/models

Roadmap


Analyse der Portierungsanforderungen
1
Analyse der Portierungsanforderungen
2
Erweiterung der Analysekapazitäten
3
Entwicklung einer interaktiven Benutzeroberfläche
4
Automatisierung des Portierungsprozesses
5
Validierung und Optimierung der Methodik
6

Timeline

Erste Auseinandersetzung mit dem Thema

Hinzufügen von weiteren Refactoring-Methoden

Hinzufügen von weiteren Analysen

Implementierung von Gemini und Pylint

Integration einer GUI

Docker Integration. Zusammenführen aller Komponenten

Danke!