

Generative KI für automatisierte Merge Request-Analyse

Guido Zockoll, Senior AI Architect bei iteratec GmbH



Welche GenAI Tools setzt ihr im Moment ein?

iteratec

JAVA FORUM
stuttgart

- ChatGPT
- GitHub Copilot
- Continue.dev
- Aider



Foto von Anastasia Shuraeva: <https://www.pexels.com/de-de/foto/werkzeuge-werkzeugkasten-nahansicht-8470252/>

Jede/Jeder hat schon mal Code mittels ChatGPT generiert.

Meist läuft der Code nicht im ersten Versuch.

- Vergessene Imports
- Nicht deklarierte Variablen
- Veraltete Libraries
- ...

Erst durch mehrfaches iterieren und manuelle Eingriffe kommt man zu lauffähigem Code.

Write the Game of Snake in Python



Sure, here's a simple implementation of the classic Snake game in Python using the `pygame` library. This code sets up the game window, handles user input, updates the snake's position, and detects collisions.

First, ensure you have the `pygame` library installed. You can install it using pip:

```
sh Code kopieren  
  
pip install pygame
```

Here's the code:

```
python Code kopieren  
  
import pygame  
import time  
import random  
  
# Initialize pygame  
pygame.init()  
  
# Define colors  
white = (255, 255, 255)  
yellow = (255, 255, 102)  
black = (0, 0, 0)  
red = (213, 50, 80)  
green = (0, 255, 0)  
blue = (50, 153, 213)
```

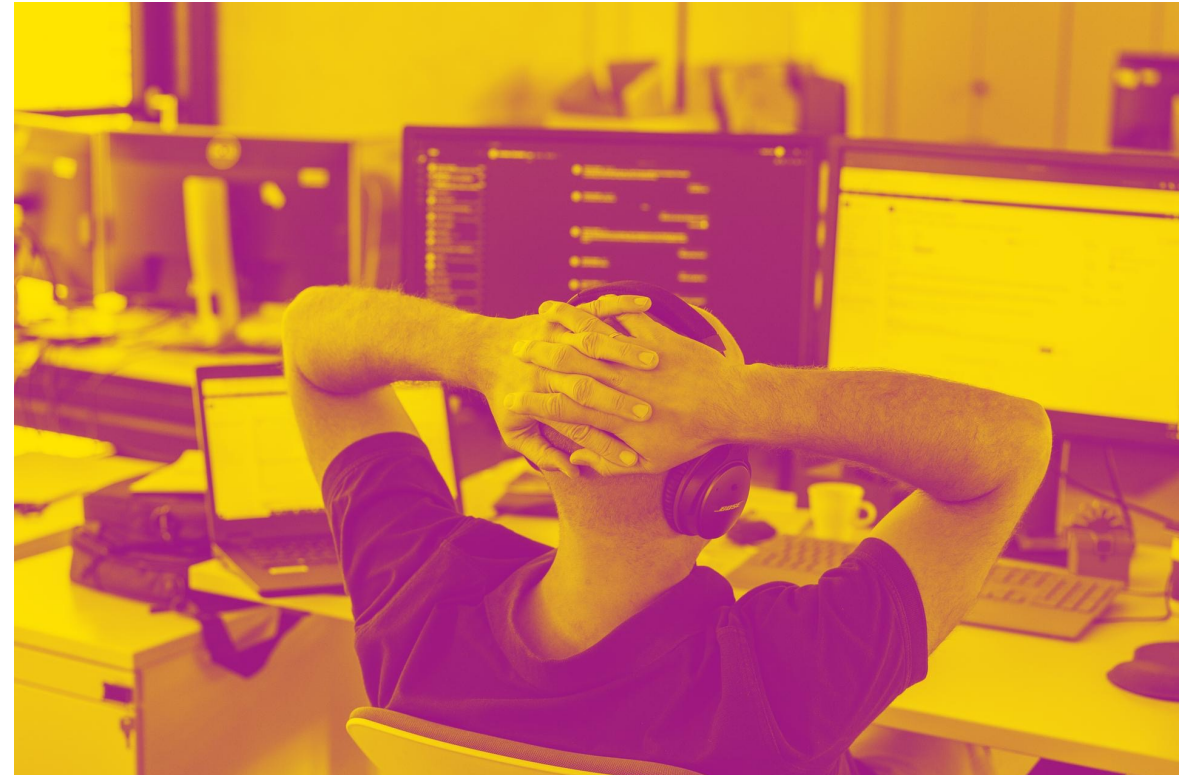


Sende eine Nachricht an ChatGPT



Aspekte der Software Qualität

- Fehlerfrei übersetzbar und ausführbar 🖱️ Compiler
- Fehlervermeidung und –erkennung 🖱️ Unit Tests
- Codequalität und Wartbarkeit 🖱️ Code Review
- Sicherheitslücken minimieren 🖱️ Code Review / statische Codeanalyse
- Leistungsoptimierung 🖱️ Code Review / Lasttests
- Kompatibilität und Interoperabilität 🖱️ Code Review / Integrationstests
- Dokumentation und Nachvollziehbarkeit 🖱️ Code Review



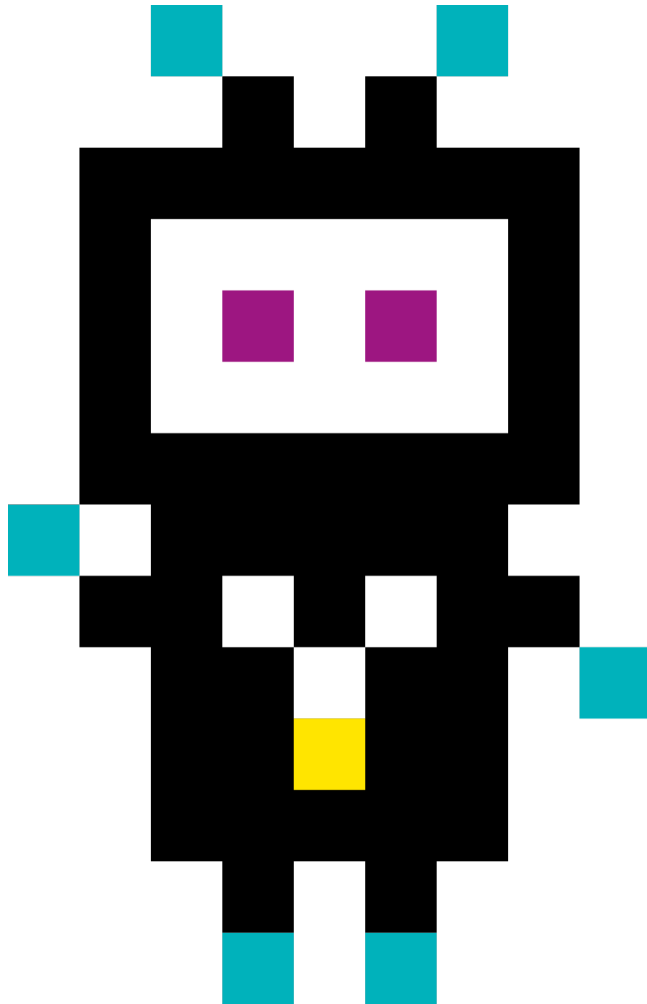


Im Software-Entwicklungsprozess werden Codereviews meist im Zusammenhang mit Merge-Requests durchgeführt.

Gerade in kleinen Teams kann man schon mal Stunden oder Tage auf ein Codereview warten. Das behindert u.U. das weitere Vorankommen. Ein Robo-Reviewer kann hier ein schnelles erstes Feedback geben. Das hat verschiedene Vorteile:

- der Autor bekommt binnen Minuten ein erstes Feedback.
- spätere Reviewer bekommen eine Zusammenfassung, was mit dem MR erreicht werden soll.

Live-Demo



Activity

All activity ▾ ↑≡



ai-coderevier @group_6719_bot_2a7e6150dc8a3edcff3d6e334925541c · 3 weeks ago

Developer



Summary of the merge request

The merge request aims to refactor the `updateQuality` method in the `GildedRose` class and introduce an `UpdateStrategy` enum for better encapsulation and extensibility of item quality update logic. The changes are intended to clean up the existing nested conditionals and provide a more maintainable structure for handling different item types. 🍷✅

Security assessment 🗝️

The provided changes do not seem to include any potential vulnerabilities or sensitive information. The refactoring is focused on the logic within the application code and does not interact with external systems or manage security-sensitive data. ✅

Error handling and code reliability evaluation 🛠️

The refactoring introduces an enum with different strategies, each responsible for updating the quality of the items. The code leverages `Math.max` and `Math.min` to enforce the quality bounds, which should prevent invalid quality values. The refactor does not appear to introduce new error handling concerns. However, there is no explicit error handling for unexpected item names that do not match any predefined strategy. The unknown items will default to the `NORMAL` strategy, which may or may not be the intended behavior. 🔍✅

Adherence check to code standards and best practices 📄

The changes seem to follow good coding practices by reducing complexity and improving readability through the use of the Strategy pattern. The enum `UpdateStrategy` provides a clear and extensible way of applying different quality update rules for different item types. The code follows Java naming conventions and encapsulates the behavior associated with each item type effectively. ✅

Einbindung in GitLab CI/CD-Pipeline:

- Konfiguration der Umgebungsvariablen im CI/CD-Setup
- Hinzufügen eines neuen Jobs in der .gitlab-ci.yml Datei

```
review_changes:
  stage: review
  cache: {}
  image: registry.iteragit.iteratec.de/taskforce-genai/ai/ai-codereviewer:develop
  script:
  - review
  variables:
  - # set the AZURE_API_KEY in the project variables/secrets in Gitlab/Github
  - GITLAB_TOKEN: $GITLAB_TOKEN
  - GITLAB_API_BASE_URL: https://iteragit.iteratec.de
  rules:
  - if: '$CI_MERGE_REQUEST_IID && $CI_MERGE_REQUEST_TITLE && $CI_MERGE_REQUEST_TITLE != "Draft: *"'
    when: always
  - when: never
  needs:
  - job: test
  dependencies:
  - test
  interruptible: true
```

Als erstes Feedback super, hat schon einige Verbesserungen gefunden

Ich finde eine Wall-of-Text immer sehr anstrengend zu lesen

Beeindruckende Zusammenfassung des MR

Wie findest du die AI-Reviews?

Hat auf Sicherheitsrisiken hingewiesen

Automatisierung an dieser Stelle ist inhaltlich und im Prozess aus meiner Perspektive (PO) eine gute und wertvolle Idee



Gute Auflistung von wichtigen Review Kriterien

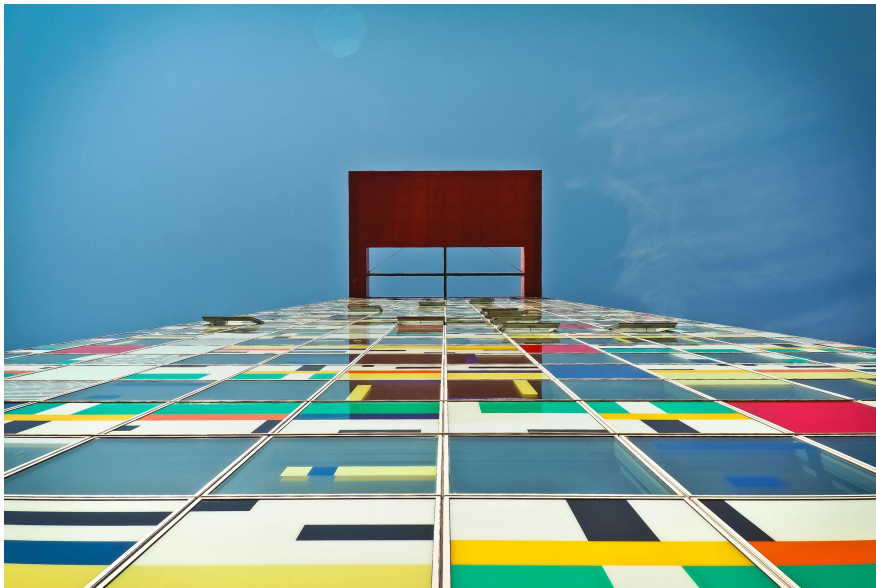
Check | Or-Not würde ich aber eher an den Anfang stellen pro "Item"

gute Idee, liefert schnell erstes Feedback (ohne dass ich Folien pushen muss :))
kann mir leider nur einmal ansehen :-(
erweiterbar und anpassbar an unterschiedliche Umfelder



Codereviews im Daily Doing

- Shell GPT Prompt
- Automatische Fixes mit Aider (<https://aider.chat/>)



```
→ cat app.py | sgpt codereviewer
Approval: needs changes

## Recommendations and Insights

1. **Import Organization**:
   - The import statements are not organized. It is a good practice to separate standard library imports, third-party imports, and local application imports with a blank line for better readability.

2. **Function Documentation**:
   - Ensure that all functions have consistent documentation. For example, the `calculate_wstf` function lacks a clear description of its purpose, parameters, and return value.

3. **Error Handling**:
   - In the `load_yaml_file` function, consider using more specific exceptions instead of generic ones. This will help in identifying the cause when debugging.
   - The `call_model` function catches all exceptions, but it would be more specific by the type of exceptions it is handling. It would be better to specify specific exceptions where possible to avoid masking unexpected errors.
```

