Herausgeber et al. (Hrsg.): Software Engineering im Unterricht der Hochschulen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 1

OPPSEE – eine Online-Plattform zum Programmieren Üben

Axel Schmolitzky[1] Henri Burau[2]

Abstract: Programmieren Können ist eine Kernkompetenz für Informatiker*innen. Um gut Programmieren zu können, ist sehr viel Programmieren Üben notwendig. Es gibt etliche Online-Plattformen, auf denen Programmieraufgaben zum Üben zur Verfügung gestellt werden. Ein Problem für Lernende auf diesen Plattformen ist häufig, die zum eigenen Lernfortschritt passenden Aufgaben zu finden. Wir stellen eine neue Plattform mit einem klaren Fokus auf das Programmieren Üben vor und beschreiben ihre flexible Architektur.

Keywords: Programmieren; Programmierlehre; Softwareentwicklung; Automatisierte Bewertung von Software; Programmierplattformen

1 Einleitung

Programmieren Können erfordert, neben weiteren Fähigkeiten, sowohl Lese- als auch Schreibkompetenz für formalisierte Texte (Quelltexte) und damit einige der Kernkompetenzen für ein erfolgreiches Informatikstudium [Be21]. Für viele Studierende der Informatik an der HAW Hamburg sind die einführenden Programmierveranstaltungen die höchsten Hürden in ihrem Curriculum, unter anderem, weil diese Veranstaltungen nicht nur mit einer klassischen Klausur abgeschlossen werden, sondern auch mit einer praktischen Programmierprüfung am Rechner [Sc17].

Programmieren Können ist keine rein intellektuelle Fähigkeit, sondern hat auch einen sehr hohen handwerklichen Anteil. Ähnlich wie bei humanoiden Muskeln kann durch regelmäßiges Trainieren des „Programmiermuskels“ eine höhere Leistungsfähigkeit erzielt werden. Deshalb sind bei den Programmierveranstaltungen im Department Informatik der HAW Hamburg die Praktika ein zentraler Bestandteil, in dem der aktive Umgang mit Quelltexten geübt und überprüft werden kann. Die Erfahrung zeigt aber auch, dass der Umfang des Praktikums bei weitem nicht ausreicht, um signifikant Programmiererfahrung sammeln zu können. Ein hoher Eigenbeitrag der Studierenden außerhalb der Präsenzveranstaltungen ist für einen erfolgreichen Abschluss notwendig.

Auf diversen Online-Plattformen wie *HackerRank* [HR22], *Codewars* [CW22] oder *edabit* [eda22] stehen zwar viele Programmieraufgaben zum Üben zur Verfügung; aber das Suchen und Finden geeigneter Aufgaben, die zum eigenen Lernfortschritt passen, ist angesichts

[1] HAW Hamburg, Dep. Informatik, Berliner Tor 7, 20099 Hamburg, Deutschland, axel.schmolitzky@haw-hamburg.de
[2] HAW Hamburg, Dep. Informatik, Berliner Tor 7, 20099 Hamburg, Deutschland, henri.burau@haw-hamburg.de

eines straff konzipierten Vollzeitstudiums von den meisten Studierenden nicht leistbar, insbesondere bei mangelnder Programmiererfahrung, wie sie in den kritischen Einstiegsveranstaltungen der ersten Semester vorherrscht. Seit 2020 entwickelt ein Projektteam des Departments Informatik an der HAW Hamburg deshalb eine Online-Plattform, auf der Studierende ergänzend zu unseren Programmierveranstaltungen mit passenden Aufgaben üben können. In diesem Artikel stellen wir die Entwurfsprinzipien und die Architektur dieser Plattform vor.

2 Grundkonzepte von Programmierübungsplattformen

Im Folgenden kürzen wir Online-Plattformen, auf denen Programmieren geübt werden kann, mit OPP ab, als Akronym für das englische *Online Programming Practice plattform*. Wir betonen den Begriff *Practice*, also Üben, um einerseits eine Abgrenzung zu allgemeinen Online-Programmierplattformen zu verdeutlichen (Online-Versionen von *Visual Studio Code* oder *Eclipse* beispielsweise sind vollwertige Online-IDEs, die aber keinen Lehranspruch erheben) und andererseits als Abgrenzung zu Online-Lernsystemen, auf denen Programmierkonzepte systematisch gelehrt werden bzw. gelernt werden können (mit ergänzenden Folien, Videos, etc.). Eine OPP bietet somit in der von uns verstandenen Reinform keine Lehrinhalte an, sondern fokussiert auf möglichst effektives Üben.

Eine OPP hat folglich als ein zentrales Konzept die *Aufgabe*. Eine Person, die auf einer Plattform eine solche Aufgabe bearbeitet, reicht einen *Lösungsversuch* in Form von Programmtext ein. Eine zentrale Anforderung an eine OPP ist, dass sie *automatisiert Feedback* zu Lösungsversuchen liefert. Bei inkorrekten Lösungsversuchen geht es um geeignetes Feedback zur statischen Übersetzbarkeit und zum dynamischen Verhalten, dies ist essenzieller Bestandteil einer OPP. Aber auch bei korrekten Lösungsversuchen kann über eine OPP Feedback gegeben werden, beispielsweise zur Qualität der gelieferten Lösung: im Vergleich zu anderen über Rankings, als Bewertung der Speicher- und/oder Laufzeiteffizienz oder schlicht als Hinweis auf alternative Lösungen, die dem System bereits bekannt sind.

Aufgaben können *sprachunabhängig* sein oder *sprachspezifisch*. Die meisten Plattformen ermöglichen sprachunabhängige Aufgaben, indem sie den Input für einen Lösungsversuch über Standard-Input liefern und den (zu bewertenden) Output auf Std-Out erwarten. Diese beiden Kanäle existieren in jeder Programmiersprache. Folglich kann beispielsweise auf der Plattform *CodingGame* [CG22] bei vielen Aufgaben aus einer Vielzahl unterstützter Programmiersprachen für den Lösungsversuch gewählt werden. Ein Nachteil solcher Aufgaben ist, dass ein teilweise beträchtlicher Anteil in das Parsing der Eingabe und die korrekte Formatierung der Ausgabe investiert werden muss. Dieser Aufwand kann bei einfacheren Aufgaben leicht die eigentlichen Lernziele „verschütten“, abgesehen davon, dass Programmieren über Std-In-Out keine Praxisrelevanz besitzt.

Wenn das Verständnis von sprachspezifischen Konzepten geübt werden soll, reicht dieser generische Ansatz auf jeden Fall nicht mehr aus. Wenn beispielsweise mit einer Aufgabe

das Verständnis der Objektstruktur zweidimensionaler Arrays in Java überprüft werden soll („Ergänze in einem gegebenen zweidimensionalen int-Array von gleichlangen Zeilen eine weitere Zeile für die Summen der Spalten, erhalte aber in der Ergebnisstruktur die gelieferten Zeilen-Arrays“), dann würde ein „Plattklopfen“ der Array-Objekte in eine zweidimensionale Textstruktur dem Lernziel nicht dienen bzw. ein Überprüfen des Erfolges unmöglich machen.

3 Das Design und der Einsatz von OPPSEE

Im Sommer 2019 hat das Department Informatik der HAW Hamburg angesichts damals noch vorhandener HSP-Mittel beschlossen, eine eigene OPP für Programmieraufgaben zu entwickeln. Den Entwurfsraum für solche Plattformen haben wir 2020 auf der SEUH in Innsbruck dargestellt [GHS20] und anschließend den Namen unserer Plattform angelehnt an den Titel der Publikation als das Akronym OPPSEE für *Online Programming Practice for Software Engineering Education* gewählt.

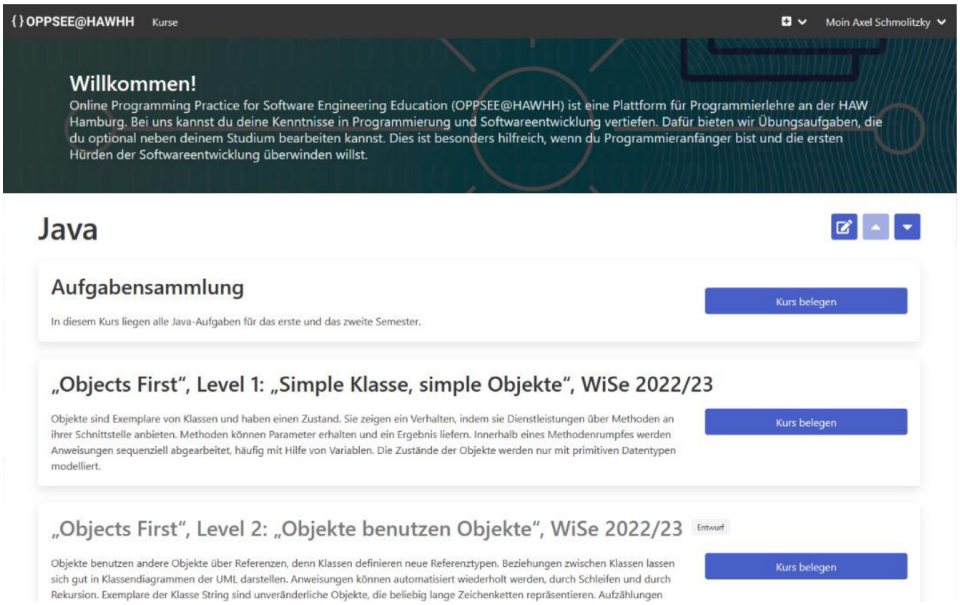

Abb. 1: Die Einstiegsseite von OPPSEE

3.1 Design

Eine zentrale Entwurfsvorgabe für OPPSEE war, dass die Plattform lediglich eine *freiwillige Ergänzung* zu bestehenden Lehrveranstaltungen darstellen, aber keine Lehre auf der

Plattform *betrieben* werden sollte. Damit ist gemeint, dass beispielsweise das Bereitstellen von Praktikumsaufgaben und die Kontrolle ihrer termingerechten Bearbeitungen explizit nicht auf der Plattform abgebildet werden sollen. Lehrveranstaltungen sollten nicht von OPPSEE abhängig sein. Auch summative Prüfungen sind auf OPPSEE derzeit bewusst nicht vorgesehen. Dies befreit den Entwurf vom Ballast aller Konzepte, die für den Nachweis der Identität und der individuellen Leistung oder das Aufspüren von Plagiaten notwendig sind. Ein weiterer Unterschied zu klassischen Lernmanagementsystemen (LMS) ist, dass wir eine weitgehende Anonymität der Benutzer anstreben. Die meisten LMS entlasten vor allem die Lehrenden, indem sie diesen Mechanismen zur Kontrolle ihrer Lernenden anbieten. Die Einstiegshürde zur Nutzung einer OPP sollte aber möglichst niedrig sein, u.a. indem ein Raum zum Üben angeboten wird, den die Studierenden angstfrei nutzen können.

Eine damit verbundene Design-Entscheidung, die den Gesamtentwurf ebenfalls erleichtert hat, war das Verwenden bestehender Authentisierungskomponenten, in diesem Fall der hochschulweiten Gitlab-Installation. Jeder Angehörige der HAW Hamburg hat eine sogenannte a-Kennung, die den Zugang zu allen digitalen Diensten ermöglicht, auch zu der Gitlab-Instanz. Benutzer können auf diese Weise sehr leichtgewichtig in OPPSEE modelliert werden und bleiben unter ihrer a-Kennungen weitgehend anonym. Dies bewirkt allerdings auch, dass die Plattform derzeit nicht ohne weiteres für die Web- Öffentlichkeit zur Verfügung gestellt werden kann.

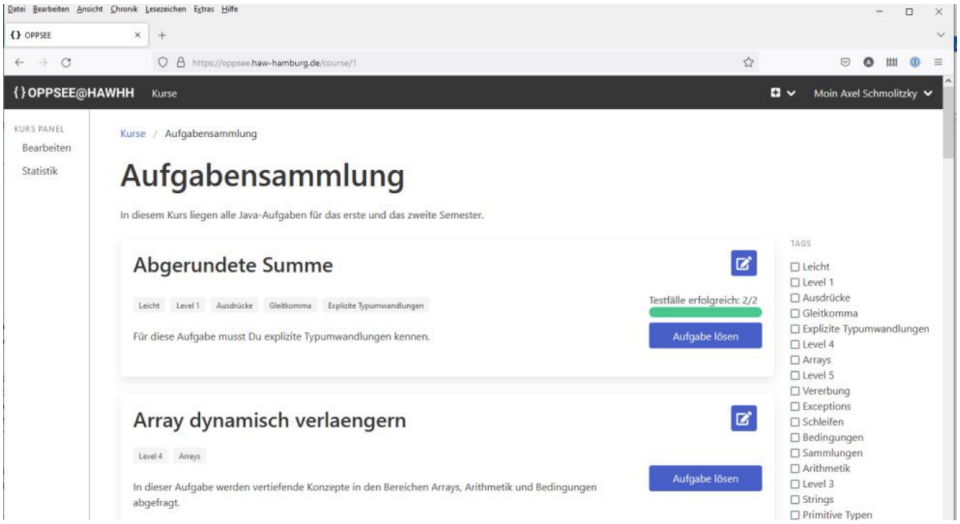

Abb. 2: Ein Kurs in OPPSEE

Eine Kernidee der Plattform ist, dass Lehrende gezielt passende Aufgaben zu ihren jeweiligen Veranstaltungen anbieten können. Sie sollen damit die Studierenden ihrer Veranstaltungen davon entlasten, sich im Web auf den diversen existierenden OPPs selbst Aufgaben suchen zu müssen, die zu ihrem aktuellen Lernfortschritt passen. Die Bündelung mehrerer Aufgaben zu diesem Zweck nennen wir auf der OPPSEE-Plattform einen Kurs. Der Begriff ist dabei leicht

irreführend, weil weder eine Anmeldung notwendig ist noch eine Zugangsbeschränkung vorgenommen wird. Jede auf OPPSEE angemeldete Person kann einen der angebotenen Kurse „belegen“ (Abb. 1) und direkt mit der Bearbeitung der verknüpften Aufgaben beginnen (Abb. 2). Die Lehrenden können die von ihnen ausgewählten Aufgaben zusätzlich mit Tags versehen, die aus Sicht des jeweiligen Kurses sinnvoll und passend sind.

Die Bearbeitung einer Aufgabe erfolgt in einer vollwertigen Online-IDE, die neben einem Editor mit Tabs, Syntax-Highlighting und Code-Completion auch einen Debugger, einen Datei-Explorer und die Anbindung an ein Git-Repository ermöglicht (Abb. 3). Der Leistungsumfang ist dabei teilweise von der unterstützten Programmiersprache abhängig. Derzeit gibt es Unterstützung für Java, C und Python. Diese Online-IDE ist keine Eigenentwicklung, es wird eine frei verfügbare IDE eingebettet (siehe nächster Abschnitt zur Architektur).

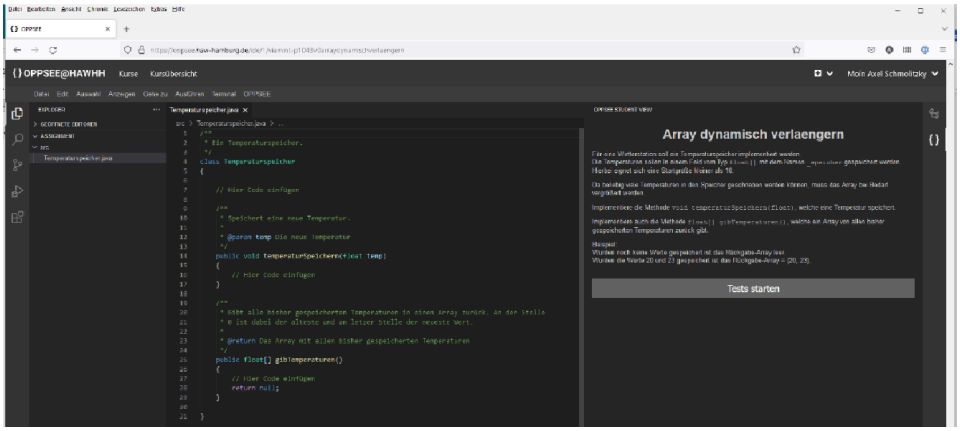

Abb. 3: Die Online-IDE in OPPSEE

Nach der erfolgreichen Bearbeitung einer Aufgabe fragt die Plattform explizit Feedback zum empfundenen Schwierigkeitsgrad, zur Verständlichkeit der Aufgabenstellung, zum Zeitaufwand und zum persönlichen Lerneffekt durch die Aufgabe ab. Die dabei erhobenen Daten werden für die Lehrenden zu Statistiken verdichtet, um die Wirksamkeit der Aufgaben besser einschätzen zu können (Abb. 4). Unter anderem setzen wir hier *Progress Networks* [Mc21] ein, mit denen Bearbeitungsversuche über die Testfälle visualisiert werden können. Diese erfordern zwar, dass es eine sinnvolle lineare Ordnung der Testfälle gibt (beispielsweise von leicht zu anspruchsvoll), liefern dann aber eine aussagekräftige Visualisierung, in der besonders häufig auftretende Abfolgen hervorgehoben sind. Auch hier liegt der Fokus auf einer Kontrolle der Aufgaben, nicht der Studierenden. Es soll keine *Leistungskontrolle* der Studierenden durch die Lehrenden implementiert werden, sondern primär für die Studierenden eine *Leistungsunterstützung*, deren Wirksamkeit für die Lehrenden ausgewertet wird.

Abb. 4: Bearbeitungsstatistik zu einer Aufgabe (Auszug)

3.2 Bisherige Einsätze

Die Plattform wurde bisher in drei Veranstaltungen eingesetzt, in denen Java vermittelt wird. Im Department Informatik waren dies eine Erst- und eine konsekutive Zweitsemesterveranstaltung zur Programmierung, im Department Informations- und Elektrotechnik eine Drittsemesterveranstaltung (dort wird in den ersten beiden Semestern C und C++ vermittelt). Ein weiterer Einsatz erfolgte in einer einführenden Veranstaltung zur Programmierung mit C im Department Wirtschaftsingenieurwesen.

Die in Abb. 1 sichtbaren Beispiele sind Kurse, die ergänzend zu einer Lehrveranstaltung nach dem Objects First-Ansatz [BK16] angeboten wurden, deren Inhalte im Sinne einer Gamification in vier Level eingeteilt sind und deren didaktische Grundlagen bereits auf der SEUH vorgestellt wurden [SZ07]. Auf jedem Level werden Programmierkonzepte eingeführt, die in den ergänzenden OPPSEE-Kursen gezielt geübt werden können. Eine Idee dieser expliziten Level ist, dass Sprachkonzepte, die auf einem späteren Level eingeführt werden, nicht für die Lösung von Aufgaben auf früheren Leveln eingesetzt werden dürfen. Beispielsweise werden Arrays als Konzept erst auf Level 4 eingeführt und sind deshalb nicht zugelassen als Lösungsbestandteil von Aufgaben auf Level 1 bis 3. Dieser grundsätzliche didaktische Ansatz, der in der Lehrveranstaltung schon seit Jahren in Vorlesung und Praktikum gelebt wird, wird derzeit noch nicht in den OPPSEE-Aufgaben technisch erzwungen. Die Plattform ermöglicht es zwar über eine mögliche Quelltextanalyse, dies ist aber bisher nicht umgesetzt.

Dies liegt unter anderem daran, dass der Aufwand für die *Erstellung* von Aufgaben für die Lehrenden vergleichsweise hoch ist. Erst wenn ein Fundus an Aufgaben existiert, wird es möglich sein, dass Lehrende nur noch passende Aufgaben für ihre Veranstaltung zusammenstellen müssen. Dieses schlichte *Zusammenstellen* von bestehenden Aufgaben

zu einem Kurs ist innerhalb der Plattform sehr komfortabel umgesetzt. Am Beispiel der Objects First-Veranstaltung wird auch deutlich, dass es zu einer Lehrveranstaltung durchaus mehrere "Kurse" geben kann. Eine weitere didaktische Idee der Einteilung in Level ist deshalb, dass Studierende am Ende eines Levels Feedback über ihren bisherigen Lernerfolg erhalten können, indem sie die Aufgaben zu diesem Level zum vertiefenden Üben bearbeiten. Idealerweise geschieht dies semesterbegleitend; allerdings zeigt die Erfahrung, dass viele Studierende doch erst kurz vor der Prüfung von dieser Möglichkeit Gebrauch machen.

4 Die Architektur von OPPSEE

Die Plattform wurde als eine Micro-Service-Architektur entworfen, um eine leicht erweiterbare und zuverlässige Anwendung zu erhalten. Dabei werden sowohl die Microservices als auch die Online-IDEs in einem Kubernetes-Cluster betrieben. Abb. 5 zeigt das Zusammenspiel der Microservices in einem Komponentendiagramm. Drei Komponenten können als Subsysteme identifiziert werden: Das Dashboard, die Online IDE und das OPPSEE Backend. Nutzer interagieren mit der Plattform über das Dashboard oder die Theia-IDE. Dabei dient das Dashboard zur Verwaltung und Auswahl der Kurse und Aufgaben, die auf OPPSEE zur Verfügung stehen. Die Bearbeitung der Aufgaben geschieht in der Online IDE. Das OPPSEE-Backend ist für das Speichern der Daten sowie für die Bereitstellung der IDEs verantwortlich. In den folgenden Abschnitten werden die Komponenten näher erläutert.

4.1 Dashboard

Als primärer Einstiegspunkt in OPPSEE dient das Dashboard. Nach der Anmeldung können Studierende dort die Aufgaben zur Bearbeitung auswählen. Die Auswahl wird dabei durch die Organisation der Aufgaben in Kategorien und Kurse, sowie durch verschiedene Filtermöglichkeiten erleichtert. Die Aufgabenerstellenden haben ihrerseits die Möglichkeit neue Aufgaben zu erstellen und diese in Kurse und Kategorien einzuteilen. Anhand der Bearbeitung und dem Feedback der Studierenden werden außerdem Statistiken erstellt und dargestellt, die Aufschluss über die Qualität der Aufgaben geben.

4.2 Online IDE

In der Online IDE erfolgt die Aufgabenbearbeitung der Studierenden. In Zukunft soll dort außerdem die Erstellung der Aufgaben möglich sein. Als Grundlage dient dabei Theia, eine IDE inspiriert von Visual Studio Code, welche komplett im Browser operiert. Klassische Funktionen einer IDE wie das Verwalten von Quellcode, Syntaxhighlighting und das Nutzen des „Language Server Procotols“ für Autovervollständigung sind bereits in Theia vorhanden. Ergänzend wurde eine Anbindung an das OPPSEE-Backend hinzugefügt.

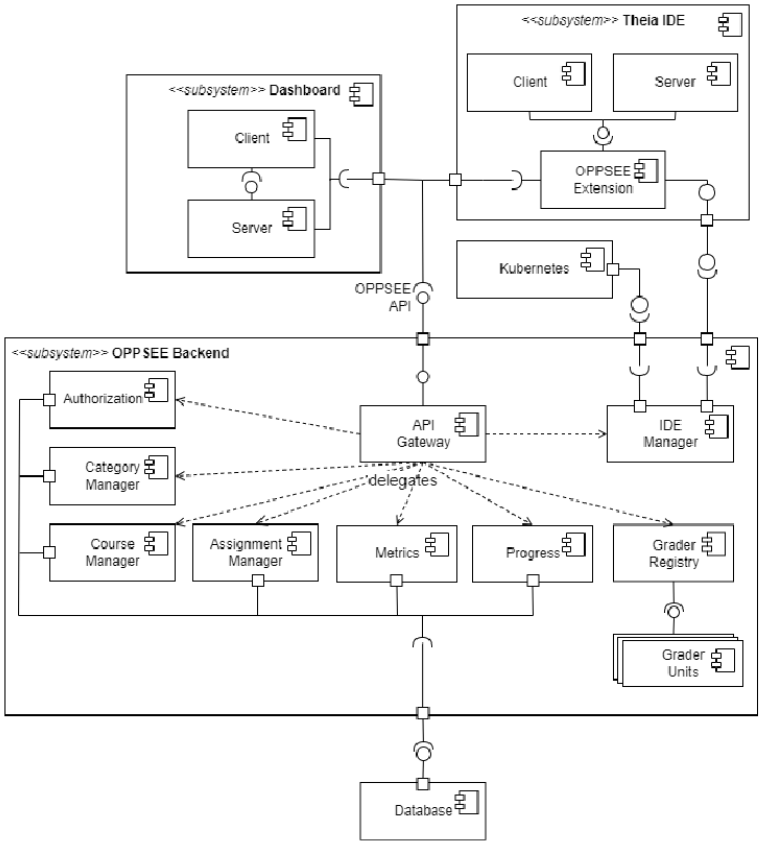

Abb. 5: Überblick der Microservice-Architektur von OPPSEE

Dadurch können Aufgaben direkt in der IDE angezeigt und zur Bewertung freigegeben werden. Auch das Feedback einer ausgewerteten Aufgabe wird direkt in der IDE dargestellt. Da die Schnittstelle für die Online IDE auch von lokalen IDEs wie Visual Studio Code oder Eclipse implementiert werden kann, besteht die Möglichkeit OPPSEE-Aufgaben auch lokal zu bearbeiten.

4.3 OPPSEE Backend

Das OPPSEE Backend stellt eine REST-Schnittstelle zur Verfügung, worüber Online-IDE und Dashboard Zugriff auf die unterschiedlichen Services erhalten. Als zentraler Einstiegspunkt dient dabei das skalierbare „API-Gateway“, welches die einzelnen Anfragen auf

korrekte Autorisierung überprüft. Die einzelnen Services registrieren dort ihre Dienstleistungen dynamisch, um eine leicht erweiterbare Architektur zu schaffen. Eine zentrale Aufgabe übernimmt der „IDE-Manager“, der die Online-IDEs verwaltet. Für jede Aufgabe und Studierenden wird eine eigene IDE gestartet, die in Form eines Kubernetes-Pods betrieben wird. Bearbeiten also 60 Studierende auf der Plattform Aufgaben, werden 60 Pods gestartet. Nach einer festgelegten Dauer von Inaktivität wird der Quelltext gesichert und der Pod heruntergefahren. Um Ladezeiten zu reduzieren und die vorhandene Infrastruktur nicht zu überlasten, wird es außerdem vermieden zwei Pods mit dem gleichen Image zu starten, stattdessen können die Aufgaben in einem bereits gestarteten Pod ausgetauscht werden. Das Erzeugen von automatischem Feedback wird durch die Grader Registry sichergestellt. Dort können sich Grader Units registrieren, die ihrerseits Feedback zu Aufgaben erstellen. Grader Units melden dabei ihren Namen sowie die unterstützten Programmiersprachen. Dieses flexible System einer Grader-Pipeline erlaubt es, eine Aufgabe von mehreren, unterschiedlichen Bewertungssystemen auswerten zu lassen, um verschiedene Aspekte des Quellcodes zu bewerten. Neben funktionalen Unit-Tests und Bewertungen der statischen Quelltext-Qualität sind beispielsweise auch Bewertungen der Laufzeit- oder Speichereffizienz wünschenswert, um Aspekte wie Nachhaltigkeit beim Programmieren zu adressieren.

5 Verwandte Systeme

Es gibt einige Systeme aus dem Hochschulkontext, die ähnliche Funktionalität zur Verfügung stellen. In diesem Abschnitt werden relevante Systeme vorgestellt.

5.1 Artemis

ArTEMiS ist ein in Java geschriebenes und an der TU München entwickeltes Open-Source „AuTomated assEssment Management System“ für interaktives Lernen. Es kombiniert Versionskontrolle und kontinuierliche Integration mit automatischer Bewertung von Programmierübungen und sofortigem Feedback [KS18]. Die Studierenden können ihre Lösungen sowohl über den Online-Editor als auch über eine lokale IDE einreichen. Die Architektur von ArTEMiS ist sehr flexibel und sprachunabhängig: Sie erlaubt jede Programmiersprache, die in eine CI-Pipeline eingebaut werden kann. ArTEMiS bietet außerdem fortgeschrittene Funktionen wie die Analyse von UML-Diagrammen. Es unterstützt auch die Verwaltung von Aufgaben für Teams. Es hat sich als robustes und flexibles System erwiesen, das in großen Kursen eingesetzt werden kann. Die Zielrichtung von Artemis ist aber eine andere als bei OPPSEE. Artemis ist explizit für das Betreiben von Lehrveranstaltungen konzipiert, indem Studierende und formativ bewertete Aufgaben samt den Deadlines ihrer Abgabe modelliert sind. Das System bietet Studierenden keine voll funktionsfähige IDE ohne eine lokale Installation und die Kenntnis von Git. Wenn ArTEMiS für freiwillige Zusatzaufgaben verwendet werden würde, wären diese Hürden für Studierende möglicherweise zu hoch, insbesondere für Programmieranfänger.

5.2 JACK

JACK ist ein Framework für modulare Benotung und Feedbackgenerierung, das an der Universität Duisburg-Essen entwickelt wurde. Der Fokus von JACK liegt auf nützlichem Feedback. Lösungsversuche können direkt in JACK hochgeladen oder über ein Plugin für Eclipse eingereicht werden. Zum Testen des eingereichten Codes wird eine breite Palette von "Checkern" angeboten, die den Code mit verschiedenen Metriken analysieren können. Dazu gehören sowohl dynamische Tests als auch statische Code-Analysen [St16]. JACK kann ohne Git-Kenntnisse verwendet werden, erfordert aber eine lokale IDE-Installation. Das Einrichten und Anpassen eines Kurses in JACK ist eine recht komplexe Aufgabe, selbst mit einer großen Anzahl von bereits verfügbaren Aufgaben, so dass hier eher die Hürde für Lehrende recht hoch ist.

5.3 Code Freak

Code FREAK (Code Feedback, Review & Evaluation Kit) [CF22] ist eine Open Source Plattform zur Bearbeitung und Feedbackgenerierung, die an der Fachhochschule Kiel genutzt wird. Code FREAK ist agnostisch gegenüber Programmiersprachen und bietet eine Online-IDE, in der die Aufgaben bearbeitet werden können. Die Aufgaben können alternativ auch durch das Hochladen von Lösungsversuchen bearbeitet werden. Durch eine Implementation des LTI Standards ist eine Anbindung an moderne „Learn Management Systems" (LMS) wie Moodle möglich.

5.4 Codeboard

Codeboard [CB22] wurde an der ETH Zürich entwickelt und dort eingesetzt. Es erlaubt das Erstellen von Online-IDEs mit vorgegebenen Dateien. Der geschriebene Quelltext kann dann automatisch ausgewertet werden. Die Online-IDE ist dabei für alle Programmiersprachen nutzbar, verzichtet aber auf die Integration des „Language Server Protocols" und damit auf viele Werkzeuge moderner IDEs, wie beispielsweise eine Autovervollständigung. Damit die Ergebnisse der Tests auch in LMS aufgenommen werden können, unterstützt Codeboard den LTI Standard. Eine Unterstützung für die Darstellung von Aufgabentexten ist nicht vorhanden.

6 Die weiteren Ziele von OPPSEE

Angesichts der seit dem Startschuss 2019 veränderten Finanzsituation an der HAW Hamburg mussten wir als Entwicklungsteam zwischenzeitlich den Schwerpunkt auf kleine Aufgaben legen, die primär für einführende Veranstaltungen geeignet sind. Auf diese Weise konnte ein

hochschulweiter Nutzen der Plattform propagiert werden, weil es in etlichen Studiengängen außerhalb des Departments Informatik Veranstaltungen zu Programmiergrundlagen gibt, die einen Einstieg mit C, Python oder Java machen und bei denen OPPSEE ergänzend hilfreich sein kann. Hier sollen die bereits bestehenden Kontakte weiter ausgebaut werden.

Angetreten waren wir jedoch mit dem Anspruch eine OPP zu entwerfen, auf der auch fortgeschrittene Konzepte der Softwareentwicklung geübt werden können: Programmieren in großen Systemen, Entwickeln im Team, Verwenden eines Repositories. Eine zentrale Herausforderung wird somit sein, Aufgaben so zu entwerfen, dass sie von ihrem Umfang innerhalb einer vorgegebenen Zeit nur im Team erfolgreich bearbeitet werden können, indem eine geeignete Modularisierung und Verteilung vorgenommen wird. Dazu werden etliche Anpassungen an der Plattform notwendig werden.

Auf etwas abstrakterer Ebene ist die Entwicklung einer "Theorie" für Programmieraufgaben interessant: Wie kann der Austausch von Aufgaben unter Lehrenden erleichtert werden? Wie kann dieselbe Aufgabe in verschiedenen didaktischen Ansätzen unterschiedlich ausgeprägt sein? Wie können Varianten einer Aufgabe geeignet modelliert werden, um Redundanzen zu vermeiden?

Sobald ausreichend Nutzungsdaten vorliegen, werden Rankings wie beispielsweise auf CodingGame für die Motivation der Benutzer interessant. Dies ist eine der Wunschvorgaben aus dem Department, die wir bisher nicht umsetzen konnten.

Eine weitere Zielrichtung ist die Entwicklung von visuellen Gradern, um ähnlich ansprechende Visualisierungen von Lösungsversuchen anzubieten wie auf CodinGame. Die Grader-Architektur ist bereits auf solche Möglichkeiten ausgerichtet.

7 Zusammenfassung

OPPSEE ist eine neue Online-Plattform zum Programmieren Üben. Sie legt explizit einen Fokus auf Aufgaben und ihre Bewertung, die als Ergänzung zu Lehrveranstaltungen eingesetzt werden können. OPPSEE soll bestehende Lehre unterstützen, aber es soll keine Lehre auf OPPSEE betrieben werden. Für Lehrende wird das Bündeln von bestehenden Aufgaben in für die eigene Veranstaltung in passende Einheiten sehr einfach gemacht. Die Architektur der Plattform kombiniert eine vollwertige Online-IDE mit einem Backend zur Aufgabenverwaltung und zur Überprüfung von Lösungsversuchen. Durch die Grader-Pipeline können sehr flexibel verschiedene Aspekte eines Lösungsversuchs überprüft werden.

Literatur

[Be21] Bender, E.; Barbas, H.; Hamann, F.; Soll, M.; Sitzmann, D.: Fähigkeiten und Kenntnisse bei Studienanfänger*innen in der Informatik: Was erwarten die

Dozent*innen? Ergebnisse einer deutschlandweiten Umfrage unter Informatik-Hochschuldozent*innen. In: 9. Fachtagung Hochschuldidaktik Informatik (HDI), Dortmund. Sep. 2021.

[BK16] Barnes, D. J.; Kölling, M.: Objects First with Java: A Practical Introduction Using BlueJ. Prentice Hall Press, USA, 2016, ISBN: 0132492660.

[CB22] Codeboard, 2022, URL: https://codeboard.io, Stand: 06. 11. 2022.

[CF22] Code FREAK, 2022, URL: https://codefreak.org, Stand: 06. 11. 2022.

[CG22] CodinGame, 2022, URL: https://www.codingame.com, Stand: 06. 11. 2022.

[CW22] Codewars, 2022, URL: https://www.codewars.com, Stand: 06. 11. 2022.

[eda22] edabit, 2022, URL: https://edabit.com, Stand: 06. 11. 2022.

[GHS20] Gandraß, N.; Hinrichs, T.; Schmolitzky, A.: Towards an Online Programming Platform Complementing Software Engineering Education. In (Krusche, S.; Wagner, S., Hrsg.): Tagungsband des 17. Workshops "Software Engineering im Unterricht der Hochschulen" 2020, Innsbruck. Bd. 2531. CEUR Workshop Proceedings, S. 27–35, Feb. 2020.

[HR22] HackerRank, 2022, URL: https://www.hackerrank.com, Stand: 06. 11. 2022.

[KS18] Krusche, S.; Seitz, A.: ArTEMiS: An Automatic Assessment Management System for Interactive Learning. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education. SIGCSE '18, Association for Computing Machinery, Baltimore, Maryland, USA, S. 284–289, 2018, ISBN: 9781450351034, URL: https://doi.org/10.1145/3159450.3159602.

[Mc21] McBroom, J.; Paassen, B.; Jeffries, B.; Koprinska, I.; Yacef, K.: Progress Networks as a Tool for Analysing Student Programming Difficulties. In: Australasian Computing Education Conference. ACE '21, Association for Computing Machinery, Virtual, SA, Australia, S. 158–167, 2021, ISBN: 9781450389761, URL: https://doi.org/10.1145/3441636.3442366.

[Sc17] Schmolitzky, A.: Zahlen, Beobachtungen und Fragen zur Programmierlehre. In (Bruegge, B.; Krusche, S., Hrsg.): Tagungsband des 15. Workshops "Software Engineering im Unterricht der Hochschulen" 2017, Hannover, 22. - 23. Februar 2017. Bd. 1790. CEUR Workshop Proceedings, CEUR-WS.org, S. 83–90, 2017, URL: http://ceur-ws.org/Vol-1790/paper10.pdf.

[St16] Striewe, M.: An architecture for modular grading and feedback generation for complex exercises. Science of Computer Programming 129/, Special issue on eLearning Software Architectures, S. 35–47, 2016, ISSN: 0167-6423.

[SZ07] Schmolitzky, A.; Züllighoven, H.: Einführung in die Softwareentwicklung - Softwaretechnik trotz Objektorientierung? In: Software Engineering im Unterricht der Hochschulen 10, Stuttgart, dpunkt-Verlag. S. 87–100, 2007.