An Architecture for the Automated Assessment of Web Programming Tasks

Lara Aubele[1], Leon Martin[1], Tobias Hirmer[1], Andreas Henrich[1]

Abstract: Automatically assessing students' solutions to programming tasks in the domain of web programming requires special means due to the characteristics of web applications. This paper proposes an architecture for a web-based learning application tailored to this domain. For the implementation of the automated assessment of programming tasks, we make use of end-to-end testing and container virtualization. This allows, in contrast to other popular approaches, the coverage of tasks that include special operations like DOM manipulations, which alter the user interface of web applications, in a way that is convenient for both students and instructors. We demonstrate the capabilities and limitations of the architecture based on two common usage scenarios.

Keywords: Web programming; Automated Assessment; End-to-end testing; Docker

1 Motivation

In both academia and industry, the interest in web applications is on the rise. Due to the characteristics of web programming, students face special challenges when they approach web application development. This includes aspects like the comparatively high number of involved languages with the most prominent ones being HTML, CSS, and JavaScript, and the interaction between scripts and the user interface on the client side. Here, JavaScript is used to alter and retrieve information about the elements present in the Document Object Model (DOM), the underlying skeleton of a web application, which is defined in HTML. This relationship impedes testing web applications as the effects of scripts exceed single units. This motivates the need for a suitable learning application that is able to automatically assess students' solutions to web programming tasks that address the aforementioned aspects. Hence, in this paper, we propose an architecture for a learning application tailored to the web programming domain. The application is envisaged to complement the blended learning concept of the chair of media informatics at the University of Bamberg[2]. In addition to a literature review, a small number of qualitative interviews[3] has been conducted beforehand to gather the requirements for such an application. The architecture and technology choices

[1] University of Bamberg, Media Informatics Group, An der Weberei 5, 96047 Bamberg, Germany lara.aubele@stud.uni-bamberg.de, {leon.martin|tobias.hirmer|andreas.henrich}@uni-bamberg.de

[2] See `https://www.uni-bamberg.de/minf` (Accessed: 04.05.2021). Major parts of the present paper are based on Lara Aubele's master's thesis written at the Chair of Media Informatics.

[3] Overall eight interviews with students and one interview with a lecturer of a web technologies course have been conducted. The interviews comprise a mixture of close-ended and open-ended questions and took about 45 minutes on average.

doi:10.18420/abp2021-2

are made with respect to them. To retain focus, this paper concentrates on aspects related to the automated assessment of web programming tasks.

The paper is organized as follows: Sect. 2 investigates popular web programming learning applications and describes technologies central for our architecture. Then, the architecture itself is introduced in Sect. 3. Subsequently, Sect. 4 demonstrates its limitations and capabilities based on two scenarios, followed by remarks on administrative aspects and future work. Finally, Sect. 5 draws a conclusion.

2 Related Work & Foundations

There are many providers that offer interactive applications for learning web programming for free. However, none of them meets all of our requirements. For instance, many applications do not support adding new tasks to their curriculum, which is an essential feature for our purposes. Furthermore, the tasks in our learning application shall be part of small but still realistic projects. Providers like Codecademy[4] teach JavaScript, HTML, and CSS on a single-file level, thus focusing on the syntax and capabilities of the languages rather than the implementation of realistic projects. Another promising candidate is IntelliJ IDEA Edu[5], the educational edition of the popular IDE specifically designed for instructor-student scenarios. While instructors are able to create own courses with tasks within realistic projects, the application has to be installed locally, though. This is contrary to the interviewees' wish that they can use the application on multiple devices with their progress and code being stored but without managing multiple installations.

Another flaw of IntelliJ IDEA Edu, Codewars[6], and some approaches described in previous work [OKP17; Sc17] is their focus on unit testing [Ru06] for checking if implemented code fragments run as expected and specified in task descriptions. However, unit tests cannot cover all operations required in full web applications. The reason for this is that the effects of operations like DOM manipulations can only be observed and thus tested when the code is executed in a Browser-like environment. Here, end-to-end (E2E) testing comes to the rescue. E2E testing is capable of testing applications that comprise multiple components by observing their overall behavior [Le16; Ts01]. Applied to web programming, E2E tests specify and test the expected state of the DOM that is defined using HTML and manipulated using JavaScript. If the DOM does not show the expected effects, the tests fail, thus providing feedback in the form of failed test cases and their descriptions.

Besides testing, some approaches [Gr17] use Docker[7], a container virtualization technology that allows running applications disregarding the underlying operating system or installed programs. For this purpose, the eponymous containers provide all necessary dependencies

[4] `https://www.codecademy.com` (Accessed: 12.05.2021)
[5] `https://www.jetbrains.com/de-de/idea-edu` (Accessed: 12.05.2021)
[6] `https://www.codewars.com` (Accessed: 12.05.2021)
[7] `https://www.docker.com` (Accessed: 12.05.2021)

such that containerized applications run the same on different machines [Be14]. In contrast to virtual machines, Docker containers virtualize the operating system and not the hardware. As a result, the code within containers is not directly executed on the host machine, thus adding an extra layer of security [Bu15][8]. Thereby, Docker allows executing code produced by users in a learning application's backend with low risk. In addition, networks of and therefore interaction between Docker containers can be set up via container ports which facilitates the implementation of modular applications. For more advanced applications, technologies like Docker in Swarm mode[9] and Kubernetes[10] provide cluster management and orchestration capabilities.

3 Concept

Regarding the learning application's architecture there are two main aspects to consider. First, the central procedure for the automatic assessment of web programming tasks has to be determined (Sect. 3.1). Afterwards, the architecture of our application is introduced, in which the testing procedure is embedded (Sect. 3.2).

3.1 Testing Procedure

We consider two options for the testing procedure, i.e., the procedure that automatically assesses the users' code. For the first option, the application generates dedicated Docker images for running and testing the user code for each task upon creation by the instructor. When a user request for testing their code arrives, a container for running the user code, a container for running the tests, and a network between them are created dynamically based on the images. The usage of separate containers for running and testing follows from the way E2E testing frameworks in the web programming domain operate, where the user code has to be executed in a browser (first container) before the e2e testing framework can access the resulting application and perform the tests (second container).

For proper execution, a `docker-compose` file is used to ensure that the container for running the user code is ready before the testing container is started. This option provides the advantage that no unused Docker containers are present in the system, thus reducing the memory usage. However, experiments showed that the time required for starting the containers on demand can exceed several tens of seconds.

The second option leverages a cluster of containers that is prepared in advance for running and testing the user code. Each node of the cluster comprises a container for running the

[8] Still, Docker is not completely secure and exploits like Docker Escape Attacks can provide the advisory with root privileges [JC17] but this is beyond the scope of this paper.
[9] `https://docs.docker.com/engine/swarm` (Accessed: 31.05.2021)
[10] `https://kubernetes.io` (Accessed: 31.05.2021)

user code and a container for running the tests connected via a static network. When a user request for testing their code arrives, the assessment task is assigned to an available node of the cluster where the user code is executed and tested. While this option is significantly faster due to the pre-built containers, the application demands more memory in idle operation compared to the first option. Additionally, the pre-built containers are less flexible and require more space because the underlying images need to contain the dependencies necessary for any of the programming tasks. Otherwise, the containers would not be equally capable of running or testing user code which is necessary for this option. For this reason, it is necessary to restart the cluster with updated images if instructors add new tasks that rely on additional dependencies.

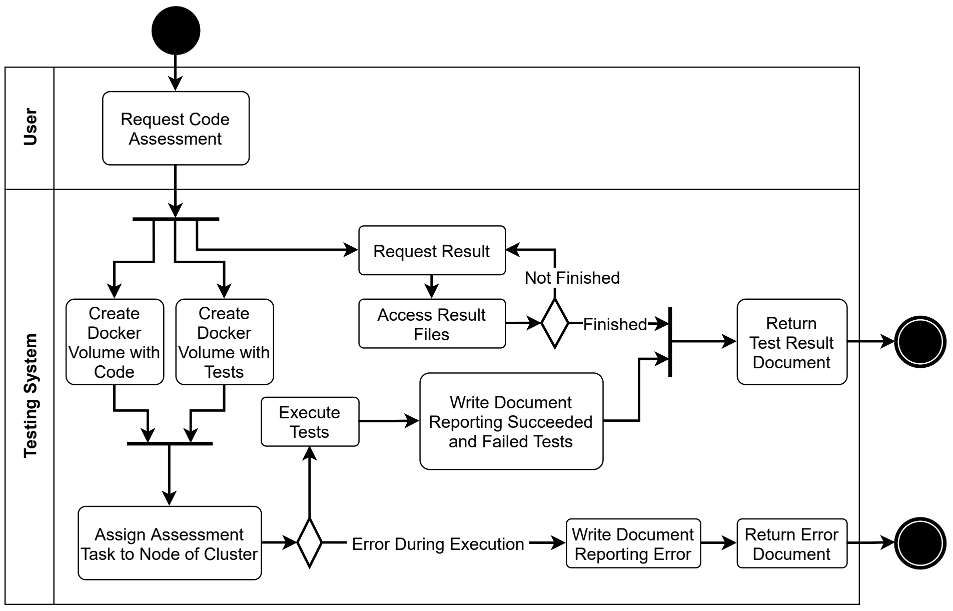
User
Testing System
Request Code Assessment
Create Docker Volume with Code
Create Docker Volume with Tests
Request Result
Access Result Files
Not Finished
Finished
Return Test Result Document
Execute Tests
Write Document Reporting Succeeded and Failed Tests
Assign Assessment Task to Node of Cluster
Error During Execution
Write Document Reporting Error
Return Error Document

Fig. 1: An activity diagram visualizing the employed testing procedure.

Considering the application's intended use and in accordance with the statements made by the interviewees, we value faster execution higher than a bigger memory footprint. We also expect that the number of simultaneous users will not exceed 25 which is comfortably manageable memory-wise[11]. As a result we settled on the second option for our application. Fig. 1 shows an activity diagram illustrating the workflow of the application's testing procedure. As depicted, the procedure has one entry point, i.e., a user requesting the

[11] According to our tests, a Docker container running a simple React app uses about 550 MiB of memory based on the `node:16` Docker image, and a Docker container executing E2E tests about 300 MiB based on the `cypress/included:3.2.0` Docker image. Note that programming tasks related to React apps are one of the most demanding tasks in terms of both web programming proficiency and memory usage that we want to cover. Even in the unlikely case that all 25 users request the assessment of their React app simultaneously, which thus represents the worst case, only about 21 GiB of memory will be used for the containers in total.

assessment of their code which triggers the testing system. Note that aside from students, instructors shall also be able to send such requests for checking the correctness of tasks and tests. Following the request, the testing procedure is traversed, eventually yielding one of two terminal states:

Internal error This terminal state occurs when the testing system itself encounters an error, e.g., a crashing container, during running or testing the user's code. Obviously, this state should never occur which motivates the usage of proper cluster management tools. In this case, a document reporting the failure is produced and returned.

Tests completed When the assigned node of the cluster finishes the tests, a document reporting the succeeded and failed tests is produced. Based on the failed test cases and their descriptions, students can improve their solution. Hence, the test cases and descriptions have to be written with high detail and care by the instructors. The document is then returned representing the other terminal state.

3.2 Architecture & Implementation

Based on the chosen testing procedure, we derived the architecture for our learning application that comprises four components. This section describes the responsibilities of each component and the technologies that we employ to implement them.

Our envisaged learning application uses several types of data. There is user data including information about the users' progress and their credentials, among others. Furthermore, there are task descriptions, task-specific code templates, and task-specific E2E tests as well as templates for Docker images. Regarding the database, no high degree of flexibility is required for the application, suggesting the usage of a standard relational database as the *Database* component. In particular, MariaDB[12], an open source SQL database, was chosen for the implementation.

For platform independence and ease of use, the learning application itself will be implemented as a web application. As to that, the *WebInterface* component naturally provides a means of interaction between the users, i.e., students and instructors, and the learning application. Its main purposes are to present the available tasks and the according task descriptions to the students, to provide an IDE where students can prepare their solutions, to allow the submissions of solutions for the automated assessment, and to display the assessment results[13]. We use the frontend library React[14] to implement the web interface. However, instead of creating our own web IDE, we leverage the Stackblitz Platform API[15] which allows embedding Stackblitz's feature-rich IDE in other web applications.

[12] `https://mariadb.org` (Accessed: 28.05.2021)

[13] There are additional features like a forum for communication between the users. However, they are beyond the scope of this paper and therefore omitted here.

[14] `https://reactjs.org` (Accessed: 28.05.2021)

[15] `https://developer.stackblitz.com/docs/platform` (Accessed: 28.05.2021)

Regarding the automated assessment of the web programming tasks, the *TaskAssessment* component represents the most important component as it houses the cluster for running and testing the user code. For setting up and managing the cluster we use Docker in Swarm mode which provides vital capabilities like load balancing and container management. This setup is compatible to the employed E2E testing framework Cypress[16]. The testing procedure itself has already been described and illustrated in Sect. 3.1.

At the application's core, the central *BackendAPI* component acts as an intermediary between the other components. Hence, it is responsible for retrieving required data from the database via an SQL interface and sharing it with other components. For communication with the WebInterface, it exposes a RESTful API implemented using Express[17]. This way, it also receives the user requests containing their code for the automated assessment.

4 Discussion

Currently, there only exists a prototypical implementation that lacks several features required for production purposes. This limits the possible depth in terms of evaluating the application and the underlying architecture. For this reason, we first walk through two common scenarios for the learning application demonstrating its capabilities and limitations, and then summarize more administrative aspects afterwards.

Scenario 1: Description As an instructor I want to create a programming task featuring a DOM manipulation so that my students can train their understanding of the interaction between JavaScript and the DOM. The task's goal shall be to implement a function that creates an `h1`-element reading *Success!* when a button is clicked on the given web page, thereby altering its DOM.

Scenario 1: Procedure The instructor first creates a project containing the solution for the task locally. Then, they write E2E tests which test if the `h1`-element is created correctly when the test triggers the `onclick`-method of the button. The test cases and their (static) descriptions[18] have to be written with high care since they represent the feedback that students will receive. Afterwards, a duplicate project is created where the parts of the code that students will have to implement are removed. All previously successful tests should fail in this project. Then, the instructor logs into the *WebInterface* using an instructor account and navigates to the administration interface where the solution, the prepared project for the students and the tests are uploaded via file upload. The *BackendAPI* receives the files via its REST API and inserts them into the *Database*. The instructor can now add a description and ascribe the task to the appropriate section before making it available to the students.

[16] `https://www.cypress.io` (Accessed: 28.05.2021)
[17] `https://expressjs.com` (Accessed: 28.05.2021)
[18] In the future, investigating dynamically generated descriptions is worthwhile to provide more helpful feedback.

Scenario 2: Description As a student I want to continue solving a task I began earlier so that I can finish the task. The said task is the task that was created in Scenario 1.

Scenario 2: Procedure The student logs into the *WebInterface* as a regular user and selects the task from the according section. After revisiting the task description, the student uses the integrated web IDE to inspect the code they already produced which is retrieved from the *Database*. They issue the automated assessment of their solution by pressing the dedicated button. The *BackendAPI* receives the request, retrieves the tests for the task from the *Database*, and relays code and tests to the *TaskAssessment*. There, an available node of the cluster runs the code and evaluates the tests. Unfortunately, one test does not succeed. Hence, a document containing the failed test cases and their description is produced and returned, first to the *BackendAPI* and subsequently to the *WebInterface* where the information is presented to the user. Now, they can iteratively improve their code and issue the assessment until all tests succeed, i.e., the task is completed. Using the description of the failed test, they realize that their code creates a `p`-element instead of an `h1`-element. Hence, they edit their code accordingly and rerun the tests yielding a report without failed tests.

From an administrative point of view, the costs of maintenance and the computational facilities represent the central drawback of our solution. The cluster-based testing procedure and the database are the main contributors in this regard. Another open aspect is that E2E testing is a black-box testing approach, i.e., the exact steps that take place within the students' solutions cannot be tested but only the effects of the code. In contrast, white-box testing like unit testing would allow assessing the code quality automatically to a certain degree. Conveniently, Cypress also supports unit and even integration testing. Hence, one major topic for future work is to integrate these capabilities in our learning application. Finally, note that the architecture proposed in this paper is just a tool that allows testing solutions to web programming tasks while the learning success heavily depends on the quality of the test cases and test descriptions.

5 Conclusion

In this paper, we introduced an architecture for a learning application tailored to web programming. We employ the E2E testing framework Cypress and Docker in Swarm mode for implementing the automated assessment of programming tasks. This enables the application to support tasks that span multiple components of a web application like DOM manipulations where JavaScript code alters the user interface. Currently, there only exists a prototypical implementation that lacks several features required for productive usage. The missing features represent future work and include the integration of unit testing as noted in Sect. 4 and aspects omitted in this paper like a forum for communication among students and instructors.

References

[Be14] Bernstein, D.: Containers and Cloud: From LXC to Docker to Kubernetes. IEEE Cloud Comput. 1/3, pp. 81–84, 2014, URL: https://doi.org/10.1109/MCC.2014.51.

[Bu15] Bui, T.: Analysis of Docker Security. CoRR abs/1501.02967/, 2015, arXiv: 1501.02967, URL: http://arxiv.org/abs/1501.02967.

[Gr17] Grummel, F.; Oechsle, R.; Brettle, A.; Schuster, D.: Automatische Bewertung von Python-Anwendungen(Automatic Evaluation of Python Applications). In (Strickroth, S.; Müller, O.; Striewe, M., eds.): Proceedings of the Third Workshop "Automatische Bewertung von Programmieraufgaben"(ABP 2017), Potsdam, Germany, October 5-6, 2017. Vol. 2015. CEUR Workshop Proceedings, CEUR-WS.org, 2017, URL: http://ceur-ws.org/Vol-2015/ABP2017%5C_paper%5C_04.pdf.

[JC17] Jian, Z.; Chen, L.: A Defense Method against Docker Escape Attack. In: Proceedings of the 2017 International Conference on Cryptography, Security and Privacy, ICCSP 2017, Wuhan, China, March 17 - 19, 2017. ACM, pp. 142–146, 2017, URL: https://doi.org/10.1145/3058060.3058085.

[Le16] Leotta, M.; Clerissi, D.; Ricca, F.; Tonella, P.: Approaches and Tools for Automated End-to-End Web Testing. Adv. Comput. 101/, pp. 193–237, 2016, URL: https://doi.org/10.1016/bs.adcom.2015.11.007.

[OKP17] Oster, N.; Kamp, M.; Philippsen, M.: AuDoscore: Automatic Grading of Java or Scala Homework. In (Strickroth, S.; Müller, O.; Striewe, M., eds.): Proceedings of the Third Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2017), Potsdam, Germany, October 5-6, 2017. Vol. 2015. CEUR Workshop Proceedings, CEUR-WS.org, 2017, URL: http://ceur-ws.org/Vol-2015/ABP2017%5C_paper%5C_01.pdf.

[Ru06] Runeson, P.: A Survey of Unit Testing Practices. IEEE Softw. 23/4, pp. 22–29, 2006, URL: https://doi.org/10.1109/MS.2006.91.

[Sc17] Schuster, D.; Brettle, A.; Oechsle, R.; Grummel, F.: Automatische Bewertung von JavaFX-Anwendungen(Automatic Evaluation of JavaFX Applications). In (Strickroth, S.; Müller, O.; Striewe, M., eds.): Proceedings of the Third Workshop "Automatische Bewertung von Programmieraufgaben"(ABP 2017), Potsdam, Germany, October 5-6, 2017. Vol. 2015. CEUR Workshop Proceedings, CEUR-WS.org, 2017, URL: http://ceur-ws.org/Vol-2015/ABP2017%5C_paper%5C_03.pdf.

[Ts01] Tsai, W.; Bai, X.; Paul, R. A.; Shao, W.; Agarwal, V.: End-To-End Integration Testing Design. In: 25th International Computer Software and Applications Conference (COMPSAC 2001), Invigorating Software Development, 8-12 October 2001, Chicago, IL, USA. IEEE Computer Society, pp. 166–171, 2001, URL: https://doi.org/10.1109/CMPSAC.2001.960613.