Teaching Complex Systems based on Microservices

Renato Cordeiro[1], Thatiane Rosa[1,2], Alfredo Goldman[1], and Eduardo Guerra[3]

[1] University of São Paulo, São Paulo, SP, Brazil
{renatocf, thatiane, gold}@ime.usp.br
[2] Federal Institute of Tocantins, Paraíso do Tocantins, TO, Brazil
[3] Free University of Bozen-Bolzano, Bolzano, Italy
guerraem@gmail.com

Abstract. Developing complex systems using microservices is a current challenge. In this paper we present our experience with teaching this subject for more than 80 students at the University of São Paulo (USP), fostering team work and simulating the industry's environment. We show it is possible to teach such advanced concepts for senior undergraduate students of Computer Science and related fields.

Keywords: Complex Systems · Microservices · Computing Education.

1 Introduction

The interest of industry and academia for the microservices architectural style increases yearly. However, its adoption is non-trivial and has many challenges. The teaching-learning process on this subject should cover relevant technical and theoretical contents. It is important to think how universities can prepare students to develop complex systems using microservices. Ideally, it should be interesting, motivating, and offer an experience close to the industry.

This paper presents our approach for teaching the development of complex systems based on microservices, as applied in the course "Laboratory of Complex Computational Systems" at the University of São Paulo (USP). Since 2018, it was offered four times as a two-week extension course with 65 students in total. In 2020, it was offered as a semester-long course with 18 students.

Our main inspiration for this course was the XP Laboratory course [2], also offered at USP. Another perspective for teaching microservices was proposed by Lange et al. [3], where students explored the subject conceptually and then focused on strangling a monolithic application.

2 About the Course

Our teaching method has three pillars: theoretical, technological and practical. The first includes lectures about complex systems, microservices and agile methodologies. The second is made of talks about front-end and back-end Web development. The third is focused on the implementation of an application based on microservices.

In the course, lectures are given by researchers and industry professionals. Students are organized into teams (4-6 members) and have to develop different domains of the system. Our assessment is continuous and incremental. The final grade is calculated based on presence and active participation in class, overtime attendance during development sprints (four extra hours per week), and the fulfillment of simple tests and exercise lists. The course also includes regular warm-up activities to foster team building and to illustrate concepts learned.

During the project development, we adopted agile practices such as peer code review, pair programming, daily meetings, and sprint and class retrospectives. To promote knowledge sharing and remote teamwork, we used Discord and GitLab.

The total course duration was 120 hours (75 theoretical and 45 practical). For a more detailed summary of the structure of the course, please access: https://uspdigital.usp.br/jupiterweb/obterDisciplina?sgldis=MAC0475.

3 Course Project

As the course project, we proposed that the students build Hacknizer: a platform to organize and host hackathons. Some of the desired features were: user access control, hackathon creation and edition, sponsor and award registration, hackathon web page customization, participant registration, team composition, project submission, and winning project choice.

The 18 students were organized into four teams, composed of 4-5 members, with different levels of knowledge and interests. We also divided Hacknizer into four bounded contexts: team management (blue). hackathon management (green), user management (red), and web page customization (yellow). Figure 1 presents an entity-relationship model that illustrates the domain of the system.

We designed Hacknizer according to a reactive microservices architectural style [1] and adopted many microservices patterns, such as Service per Team, API Gateway, Microservice Chassis, Single Database per Service, Event Sourcing, CQRS, and Saga. Figure 2 presents an architecture overview of the system.

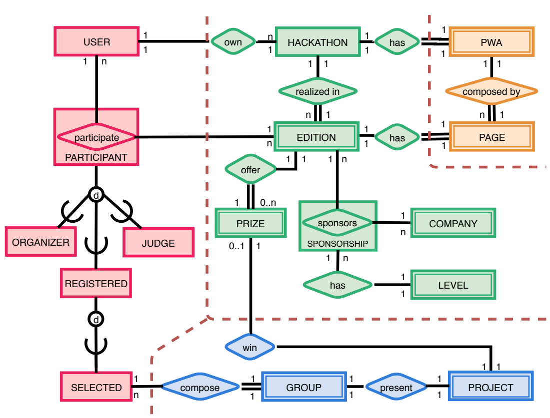

Fig. 1. Entity–Relationship Model

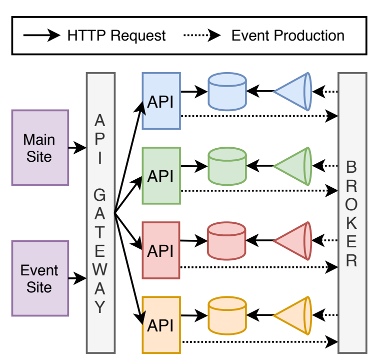

Fig. 2. Architecture

4 Lessons Learned

To evaluate the learning process, we surveyed our current 18 students to collect their self-assessed level of knowledge in 18 skills the course focuses on. Our results describe five different moments: before the course, after the first lecture block, after the first sprint (both focused on theory and Web front-end), after the second lecture block, and after the second sprint (both focused on Web back-end). Figure 3 shows a heatmap summarizing the students' answers for the 18 skills and illustrates their evolution throughout the course.

Regarding the microservices architectural style and its patterns, students with low level of knowledge before the course (1, 4-5, 9, 12-13 and 15) managed to evolve as the course progressed. On the other hand, students who had a high level of knowledge about the subject (3, 6 and 14) maintained this level.

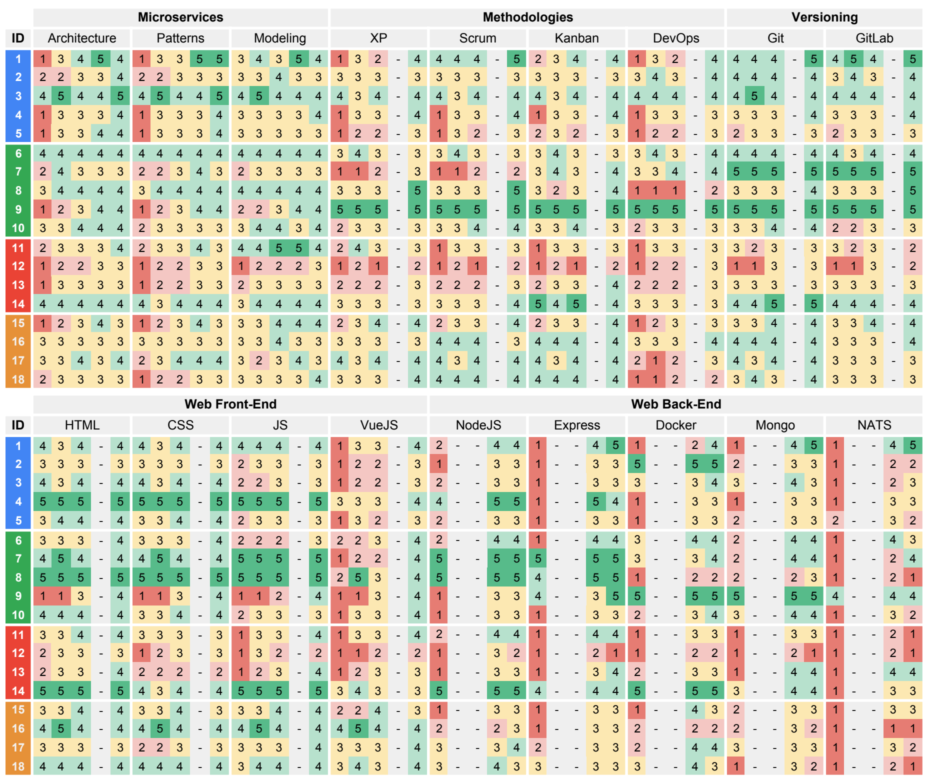

Fig. 3. Heatmap with students' self-assessed level of knowledge in 18 skills the course aims to improve. Each answer follows a Likert scale ranged 1-5, where 1 means "very low" and 5 means "very high". Each topic has answers about up to five moments: before the course, after the first lecture block, after the first sprint (both focused on theory and Web front-end), after the second lecture block, and after the second sprint (both focused on Web back-end). Dashes in the column represent a moment when the students were not asked about this subject.

With respect to agile methodologies, students who had attended the XP Laboratory course had a greater background about these subjects. During our course, they either kept or evolved their level of knowledge. Notwithstanding, all students with no previous experience in agility advanced their knowledge.

Regarding versioning, students who are not enrolled in the Bachelor's degree in Computer Science (5, 10 and 12) had a low level of knowledge in this subject before the course.

With respect to front-end technologies, most students had experience with the basic stack: HTML, CSS and JavaScript. The most unfamiliar tool was VueJS, a modern single-page application framework. Despite the challenges, even students with very low level of knowledge (9, 11 and 13) could get a good understading of how to develop with it.

Regarding back-end technologies, students who had little experience with Docker (1, 4-5, 8, 10-13 and 15-17) advanced their knowledge. Two thirds of the students had low level of knowledge about NodeJS and Express, but they assessed that they progressed to medium or high level of knowledge by the end of the course. Lastly, very few students knew how to use MongoDB or NATS. In the last survey, most of them had a good advance with the former but the same did not happen uniformly with the latter.

Besides the surveys, we made a non-structured interview with the students to ask eight questions covering their backgrounds, difficulties on each of pillar of our teaching method (theoretical, technological, practical), and general impressions.

Overall, the course is meeting the students' initial expectations. The biggest challenges reported include: team work, which is not commonly applied in other courses; and remote collaboration, since no single tool worked seamlessly for all students in all environments. The main knowledge gains include: team work, because they felt they were learning how to develop together and were enjoying the collaborative discussions that came from it; and an environment similar to the industry's, since they reported our course was the closest to the challenges they expect to deal with in a full-time job, in particular the use of microservices.

Acknowledgment

We want to thank our partner João Francisco Lino Daniel who helped to plan and develop the course offerings described in this paper. We also thank Alceste Scalas, Filipe Correia, and Rebecca Wirfs-Brock for the suggestions for improving this extended abstract.

References

1. Bonér, J.: Reactive Microservices Architecture: Design Principles for Distributed Systems (2016)
2. Goldman, A., Santos, V.A.: Continuous improvement of an xp laboratory course: An 18 year history. In: 2019 Agile Conference (2019)
3. Lange, M., Koschel, A., Hausotter, A.: Microservices in higher education. International Conference on Microservices (2019)