

APPENDIX A – DETAILED TEXTUAL DESCRIPTION OF THE PHASES OF THE ROLE-PLAY MODEL

- **Phase #1:** aims to explore the category of Knowledge Sharing. Bartol e Srivastava (2002) define knowledge sharing as the sharing of organizationally relevant information, ideas, suggestions and experiences between individuals and others, and state that knowledge sharing is a key component of knowledge management systems. In this case, the orientation of concepts manifests itself in the explanations of Collaborative culture and Sharing knowledge. In the Culture of collaboration, four points are used: prioritizing relationships, communication, aligning responsibilities/incentives and respect. Sharing knowledge involves the applicability of explicit and tacit knowledge combined with pair programming. The respective practice of concepts will take place through the division of groups of students, where already within the dynamics they must first introduce themselves, speaking their names, positions and training. After this brief presentation, a "stand-up meeting" will be held to discuss the initial items requested by the PO. This will ensure that the participants have their responsibilities aligned and maintain respect for personal and technical issues. Next, regarding explicit and tacit knowledge, the relevant API information, such as the fork, should be provided on the slides and by the tech lead. For tacit knowledge, pair programming should be used. As objectives for this phase, the participants are expected to have understood the points requested by the PO, to have made the appropriate presentations to their peers about who you are as a collaborator, to have made the divisions for pair programming and to have carried out the fork of the project and analyzed the code base. Following on from phase one, Source Code Management (SCM) is used to track modifications to a source code repository. SCM tracks a running history of changes to a code base and helps resolve conflicts by merging updates from multiple contributors. SCM is also synonymous with version control (VCS) (ATLASSIAN, 2023). The aforementioned category has version control in its orientation of concepts, which in turn has the following benefits: creating workflows, working with versions, coding together, keeping a history and automating tasks (MICROSOFT, 2023). The practice adhering to the VCS will take place as follows, with the project duly downloaded, the participants, using pair programming, must now download two packages, the first is Nodemon which is a library that will help in the development of systems with NodeJs, making it possible to automatically restart the server and the second is Express which is a framework that

provides minimal resources for building web servers. Continuing the practice, participants must complete the demands requested by the PO, but after completing each API method, two conventions must be followed, the first is Semantic Versioning (SemVer) and the other is the commits convention. For example, after completing the Get method, the team must update the version of the project in the package.json according to the respective Major, Minor and Patch of SemVer and when sending the change to the repository, they must follow the appropriate commit pattern, which can be indicated by the tech lead if there is any doubt as to which one should be used. And this same process is repeated for each method implemented. This phase is expected to achieve the following objectives: sharing knowledge among peers in the creation of API methods, in the generated package.json file, there is a "version", adapting it to the SemVer standard and having done all the commit and push to the forked repository on GitHub.

- **The Phase #2:** aims to address the Build Process (BC) category. The CP is the process of creating a complete, executable system by building and linking system components, external libraries, configuration files etc(SOMMERVILLE, 2011). We considered not only tools that generate deployable packages, also called builds, but also tools that generate essential artifacts and feedback using the source code as input (LEITE *et al.*, 2019). The concept orientation of the current phase groups together explanations of three concepts: Release Engineering (RE), Test Automation (TA) and Static Analysis (SA). In due order, RE has requirements in its theoretical foundations on certain topics, such as package management. AT uses tools and scripts for executing (MAILEWA *et al.*, 2015). These tests aim to repeat predefined actions by comparing the requirements with the actual test result (BHATT, 2017). Finally, the SAE of a program is that performed without executing it (WICHMANN *et al.*, 1995; EGELE *et al.*, 2008). Static analysis techniques can be used to check the specification and design models of a system to find errors before an executable version of the system is available (SOMMERVILLE, 2011). The practice designed for this phase begins with the installation of some packages through npm, the package manager for NodeJs, including mocha, which is a JavaScript testing framework. After installation, participants will need to create the necessary test cases and run them, receiving a positive or negative response. The first is Prettier, which is a code formatter, and the other is ESLint, which is a linting tool developed specifically for JavaScript. Immediately you will see a series of errors in the code and to solve them the students must edit some files provided

by the RP driver. For the objectives of the third phase, the students are expected to have installed all the packages provided for building the test cases, to have built the automated test file and to have succeeded in all the cases, and to have installed and configured Prettier and ESLint. Now on to the Continuous Integration (CI) category. Fowler (2006) reports that CI is a software development practice where team members integrate their work frequently. Each integration is checked by an automated build (including tests) to detect integration errors as quickly as possible. The orientation of concepts relating to the CI phase deals with the process of frequent and reliable release. The Humble e Farley (2010) establishes some points that guarantee a frequent and reliable release process, among them creating a repeatable and reliable process for delivering software, automating everything possible, keeping everything in a version control system and if a step causes pain, perform it more often and as soon as possible. The CI practical part will be implemented with the help of the GitHub Actions platform, which helps to build CI/CD pipelines. Within the repository that was forked by the participants there is a tab called Actions, after selecting it the participants will be directed to another page containing Workflows, where a respective NodeJs has to be chosen, after which they will be shown an initial code YAML that must be explained by the Tech Lead and supplemented to meet the required needs. Once the explanations have been given and the necessary changes have been made, the participants simply have to execute a commit and the CI pipelines that have been configured will be triggered. For the objectives of this phase, it is enough for the participants to have managed to build the CI pipeline and run it with a hundred percent success rate. At the end of this phase, Deployment Automation (DA) is what allows software to be deployed in test and production environments with just one click. Automation is essential to reduce the risk of production deployments. It is also essential for providing quick feedback on software quality, allowing teams to perform comprehensive testing as soon as possible after changes are made. Talking about Deployment Automation is talking about Continuous Deployment (CD), CD is the next stage after continuous delivery, where you're not just constantly creating a deployable package, but actually deploying it constantly (PAIR, 2023). For Valente (2020) the advantages of CD are that it reduces the delivery time of new features, makes new releases (or deployments) a "non-event", reduces the stress caused by deadlines and new features go into production quickly. For the practice of this phase, Fly.io will be used, which is a global application distribution platform. The first step is to install the

solution via the terminal, the second step is to register on the platform. Once these two steps have been completed, the tech lead will instruct the participants that some steps must be done via the command line via the terminal. Once the instructions have been completed, it is necessary to create a secret key which will allow the inclusion of another piece of code yml which is responsible for completing the pipeline of the CD. The objectives that must be achieved in this phase are related to having followed all the steps to create the account on Fly.io until the API is deployed on it, having added the necessary scripts to our yml file for the appropriate CD configurations and performing another commit to trigger the CI/CD pipeline and see the API deployment on Fly.io.

- **Phase #3:** The Monitoring and Registration category is discussed. O monitoramento geralmente é feito através de ferramentas que rastreiam as propriedades não funcionais das aplicações, como desempenho, disponibilidade, escalabilidade, resiliência e confiabilidade, e entre as diversas tarefas executadas pelas ferramentas de monitoramento estão Você constrói, você executa, monitoramento contínuo, métricas e gerenciamento de logs, (LEITE *et al.*, 2019). The orientation of concepts present in the Monitoring and Logging phase addresses the definition of You built it, you run it. This term was coined in 2006 by Werner Vogels, Amazon's CTO. It puts developers in touch with the daily operation of their programs. It also puts them in daily contact with the customer. This cycle of customer feedback is essential for improving the quality of the service (O'HANLON, 2006). Another point covered in the guidance is Continuous Monitoring (CM), where CM in DevOps is the process of identifying threats to the security and compliance rules of a software development cycle and architecture (SANYAL, 2022). In addition to CM, there are Metrics, which are a set of measures of a software process or product, where a software product can be seen as an abstract object that evolves from an initial instruction to the finished software system, including the source code and various forms of documentation produced during development. (MILLS, 1988). Lastly, in the guidelines there are the logs, which for Du e Li (2016) is a record of events, in a computer environment, it is the documentation produced automatically or initiated from a system or host, with the date and time of events relevant to a particular system. In the work of (CARNEIRO, 2022), he reports that the name of the host, type of log, IP address and MAC address can also be included in the logs. The practice related to this phase begins with the installation of the Sonarqube image. After installation, the container must be run and access localhost:9000

so that the first access can be made and the admin can be configured. After that, you should download and install SonarScanner, which is nothing more than a generic way of evaluating application code. Returning to Sonar, you should create a manual project, name it, choose the type of code that will be analyzed, after which a token will be generated that should be used in the terminal indicating the relative path of the project and when executed, it will cause the Scanner to analyze the files and upload them to Sonarqube. The other practice is to manipulate the body-parser and morgan-body packages, then the participants must apply the constants to the NodeJs code responsible for the API requests, after which they must create a folder in the root of the project called logs where the log file is located, making it possible to view some of the information that was recorded when the endpoints were called. The objectives assigned to the sixth phase are the correct deployment of the packages, as well as obtaining a log file containing initial information about the type of request and the data trafficked through them, as well as any call errors contained in the application.