ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/350451048

FGPE Gamification Service: A GraphQL Service to Gamify Online Education

Chapter · March 2021

DOI: 10.1007/978-3-030-72654-6_46

CITATIONS
3

READS
75

6 authors, including:

José Carlos Paiva
Institute for Systems and Computer Engineering, Technology and Science (INESC …
23 PUBLICATIONS **161** CITATIONS
SEE PROFILE

Alicja Har
Aalborg University
2 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Ricardo Queiros
Polytechnic Institute of Porto
113 PUBLICATIONS **633** CITATIONS
SEE PROFILE

José Paulo Leal
University of Porto
133 PUBLICATIONS **1,115** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Written Language Bursts View project

BalticMuseums - Digitalization in Museums View project

All content following this page was uploaded by José Carlos Paiva on 18 January 2023.

The user has requested enhancement of the downloaded file.

FGPE Gamification Service: A GraphQL Service to Gamify Online Education

José Carlos Paiva[1,3(✉)], Alicja Haraszczuk[2], Ricardo Queirós[1,4], José Paulo Leal[1,3], Jakub Swacha[5], and Sokol Kosta[2]

[1] CRACS – INESC TEC, Porto, Portugal
jose.c.paiva@inesctec.pt
[2] Department of Electronic Systems, Aalborg University, Aalborg, Denmark
aharas18@student.aau.dk, sok@es.aau.dk
[3] DCC – FCUP, University of Porto, Porto, Portugal
zp@dcc.fc.up.pt
[4] uniMAD – ESMAD, Polytechnic of Porto, Porto, Portugal
ricardoqueiros@esmad.ipp.pt
[5] Department of IT in Management, University of Szczecin, Szczecin, Poland
jakub.swacha@usz.edu.pl

Abstract. Keeping students engaged while learning programming is becoming more and more imperative. Of the several proposed techniques, gamification is presumably the most widely studied and has already proven as an effective means to engage students. However, there is a complete lack of public and customizable solutions to gamified programming education that can be reused with personalized rules and learning material. FGPE Gamification Service (FGPE GS) is an open-source GraphQL service that transforms a package containing the gamification layer – adhering to a dedicated open-source language, GEdIL – into a game. The game provides students with a gamified experience leveraging on the automatically-assessable activities referenced by the challenges. This paper presents FGPE GS, its architecture, data model, and validation.

Keywords: Gamification · Programming education · Gamification service

1 Introduction

The ever-increasing demand to adopt digital solutions became incontestable in 2020 due to the coronavirus pandemic crisis. Industries all over the world have been forced to go digital [1]. Recent studies indicate that among the fifteen most in-demand careers at the moment, three require programming capabilities, while several expect other ICT skills. Furthermore, those involving coding skills belong to the top-five best paid of the fifteen [6]. The existing difficulties in teaching and learning programming, however, only worsened [2] as educational institutions are now massively pushing towards online learning and keeping students engaged at distance is even more challenging [11].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2021
Á. Rocha et al. (Eds.): WorldCIST 2021, AISC 1368, pp. 480–489, 2021.
https://doi.org/10.1007/978-3-030-72654-6_46

Gamification, consisting of the use of game elements and game-thinking in non-game contexts, is one of the most prestigious technology-powered techniques for human engagement. Thus, to no surprise, it has been studied and utilized in the realm of programming education with evidence of success [12]. However, adopting such methodologies in a specific subject or course requires considerable effort, which often hinders its implementation. In our perspective, this is mainly due to the closedness of existing solutions, which force one to either use existing gamified platforms designed for specific needs or develop one from scratch. Currently, no platform nor service can handle a set of custom gamification rules applying them on a collection of programming exercises, much less an open repository of gamified programming exercises, so that one could reuse them in diverse contexts and adjust for distinct needs.

Capitalizing on GEdIL [13], a language designed for the representation of the set of gamification concepts and rules decoupled from educational content, the contribution presented in this paper is an open-source GraphQL service, namely the FGPE Gamification Service, that consumes a GEdIL package and applies the described layer over the automatically assessable activities referenced by its challenges. This service supports any type of activity, only requiring a plugin that registers itself as an evaluation engine of that activity (i.e., it provides automatic assessment). Currently, it ships with a plugin for Mooshak 2 [8], a web system with support for assessment in computer science.

The rest of this paper is organized as follows. Section 2 reviews existing gamification services. Section 3 presents the FGPE Gamification Service and the GraphQL gamification service, which represent the novel contribution proposed in this work. Section 4 describes the evaluation of correctness and performance of the service. Finally, Sect. 5 summarizes the contributions of this work and indicates some future directions.

2 Related Work

Entertainment, addiction, competition, or cooperation are all factors that cause people to play games. Most of these factors are achieved through the inclusion of elements and mechanics in the game design which fosters the motivation of the player extrinsically and intrinsically and, at the same time, enhances the player's retention and engagement. The interest in making those elements reusable and exploring their effects in different contexts led to their externalization and, ultimately, availability as a service. Currently, there are several options to bring these elements into (programming) education, from consuming Game-Backend-as-a-Service (GBaaS) to business-directed SaaS and full-fledged programming learning platforms, but none sufficiently flexible to adapt to a specific course and set of needs.

A GBaaS can be defined as a subset of a more general backend that includes several services specifically tailored for game development. These services allow developers to focus on the game logic while freeing them from implementing boilerplate features. Some notable examples of GBaaS implementations include

Google Play Game Services (GPGS), Yahoo Backend Game Service, GameUp, Flox, GameSparks, Fresvii, Kumakore, and Photon, which although different in their specifics, share several features available through a cross-platform API. A comparative study of the platforms [3] states that most of these features can be organized into two groups: social and technical. Social game features aim to make game-play more competitive and collaborative while improving social engagement, including leaderboards, achievements, multiplayer, quests, gifts, and matchmaking. Concerning technical game features, the majority of the GBaaS offer an API and multi-language SDK for the most popular mobile platforms (e.g., Android, iOS, FirefoxOS) and game engines (e.g., Unity, Unreal, Marmalade, Cocos2D). Both, the APIs and SDKs, support authentication where players either should have an account on specific backends (e.g., GPGS requires users to have a Google account) or provide a simple mechanism that allows games to implement social login without any additional code (e.g., GameSparks). In terms of API characterization, most of the GBaaS implementations provide an HTTP RESTful API with valid JSON payloads and responses. However, as GBaaS typically rely on external authentication, they are not adequate for e-learning systems that already operate on a single sign-on ecosystem. For this reason, some solutions that mimic them but developed for the requirements of e-learning systems also exist [10].

Considering gamification platforms allowing simple customization, they are mainly proprietary business-oriented solutions targeted to client and employee engagement, such as Gametize, GetBadges.io, and Bunchball, which add complementary features to adapt to a specific business (e.g., content management system, customized rewards, mechanisms to promote social behaviors, on-boarding, reports, and analytics). Fleeing a bit to this niche, GameLayer is a gamification and rewards platform whose main goal is to create user engagement and consumer loyalty with cloud-based game mechanics. It includes a REST API that exposes several features such as leaderboards, achievements, missions, points, prizes, mystery boxes, teams, and level-ups.

Finally, several learning platforms capitalize on gamification as a means to engage learners, providing little to no customization of its logic. For example, Khan Academy and Codecademy use badges and progress tracking to engage students in published courses. CodinGame, a web-based platform that proposes several puzzles for learners to practice their coding skills, has a larger investment on game elements, implementing graphical game-like feedback, leaderboards, rewards, points, social interaction, among others. Some Learning Management Systems also evolved to provide more engaging environments resorting to badges, achievements, points, and leaderboards such as Academy LMS, Moodle (through several plugins), and Matrix.

3 FGPE Gamification Service

FGPE Gamification Service (FGPE GS) is a GraphQL [5] service designed for the gamification of educational contexts, which transforms a layer, adhering

to Gamified Education Interoperability Language (GEdIL) [13], into a game. Having complete support for any layer described in GEdIL, including rewarding mechanisms such as points, badges, virtual items, and social status (e.g., through leaderboards), unlockable and secret content, different activity modes (e.g., speedup and duels), among others, FGPE GS only requires mapped activities to be automatically assessable.

The next subsections provide an in-depth overview of the FGPE GS. Subsection 3.1 details the API and architecture of the service. Subsection 3.2 presents its data model.

3.1 Architecture

FGPE Gamification Service is a web service developed in NodeJS using TypeScript, powered by NestJS [9] – a framework for building efficient, scalable Node.js server-side applications that combines elements from Object Oriented Programming, Functional Programming, and Functional Reactive Programming. This web service provides a GraphQL [5] API, which means that a single endpoint provides access to the complete schema as a graph. To query this graph, clients should use a special query language that allows them to specify precisely the fields they demand in the JSON response from the server, both from the target resource or any of its nested (deep) relations. For instance, with a single query, as presented in Fig. 1, it is possible to obtain a player's profile as well as the name of the granted rewards in a game.

```
{
    profileInGame(gameId: "G1") {
        game {
            name
        }
        user {
            username
        }
        rewards {
            reward {
                name
            }
        }
    }
}
```

```
{
    "data": {
        "profileInGame": {
            "game": {
                "name": "Local championship"
            },
            "user": {
                "username": "student_fgpe"
            },
            "rewards": [{
                "reward": {
                    "name": "Hardworker"
                }
            }]
        }
    }
}
```

Fig. 1. GraphQL query for player's profile (left) and respective JSON response (right)

Although GraphQL is a query language, it also provides a mechanism to modify server's data, known as mutation. In fact, the only difference between a mutation and a query is that it should be clearly indicated as a mutation, by starting with `mutation` keyword. An exception are upload requests which

should be sent as `multipart/form-data`, passing the mutation in a field named `operations` and files in fields with the name of the variables used in the mutation. Furthermore, FGPE GS also implements GraphQL subscriptions, allowing clients to register themselves as listeners of particular events, to be notified in real-time about new objects, updated fields, among others, without explicitly querying the server. Figure 2 shows an example subscription to receive notifications about granted rewards.

```
subscription {
    rewardReceived {
        name
        kind
    }
}
```

```
{
    "data": {
        "rewards": [{
            "name": "Hardworker",
            "kind": "BADGE"
        }]
    }
}
```

Fig. 2. GraphQL subscription for player's rewards (left) and respective JSON notification (right)

Once a request reaches FGPE GS, the JSON Web Token (JWT) is extracted from the cookies and validated against Keycloak [7], an open-source identity and access management solution providing centralized user management, authentication, single sign-on and identity brokering, user federation and social login, among other features for the applications. Keycloak either throws an error, in which case an HTTP 401 Unauthorized error is sent back to the client, or returns the information of the authenticated user, in which case the execution proceeds by checking the roles. There are 3 roles: `author`, who is responsible for importing GEdIL packages and manage the layer data; `teacher`, who manages assigned games, groups, and students; and `student`, who actually plays the games. For instance, to submit an attempt to solve an activity, illustrated in Fig. 3, the user must have the `student` role (and be enrolled in the game), otherwise the client receives an HTTP 403 Forbidden error.

Continuing with the case of a submission request (i.e., the main action that a player performs), after the authorization check end, the submission is received and stored in the service database, provided by MongoDB, a general-purpose NoSQL document-based and distributed database. Then, the submission is inserted into a job queue based on Redis that holds all submissions waiting evaluation. At this point, the client receives a response containing the ID of the stored submission (and other data), which can be later used to fetch the evaluation result. Meanwhile, an evaluation engine plugin, which is a consumer of these jobs and has the single responsibility of evaluating a submission and pushing the result to another queue, picks and evaluates the submission. This way of using evaluation engines allows to both replace them seamlessly and use several at the same time. Once the submission is evaluated, the stored submission is patched with the result, the respective hooks are triggered, and the result is sent to the client if it has an active GraphQL subscription.

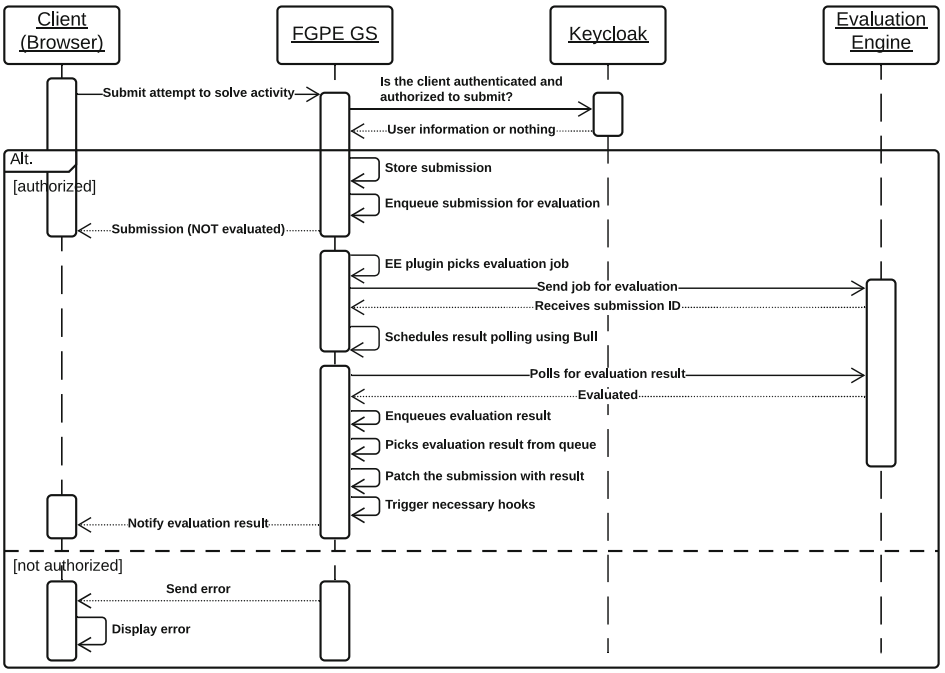

Fig. 3. Main interactions between FGPE GS components in a submission request.

Even though the FGPE GS can integrate any architecture where Keycloak handles user authentication and authorization, it was initially designed to integrate the ecosystem of the Framework for Gamified Programming Education (FGPE). FGPE GS is the core of this ecosystem, as it fetches the educational content, intermediates the automatic assessment of the activities, dictates the order and way of presenting it, and feeds the Learning Environment with the educational content, in the order and way it dictates. Due to the requirements of this ecosystem, FGPE GS includes out-of-the-box an evaluation engine plugin for Mooshak 2 [8], a web system with support for assessment in computer science.

3.2 Data Model

The core of the FGPE GS data model, presented in the entity-relationship diagram of Fig. 4, is the `Game` document. Each time an author imports an instance of a GEdIL layer through the API, the FGPE GS creates a `Game` holding a tree of `Challenge`s, `Reward`s, `Leaderboard`s, and `Hook`s, resulting from the layer conversion. Users may enroll in a `Game` or wait for one of the instructors (i.e., the teachers who will manage the game and players) to add them as `Player`s, who are then able to make `Submission`s, complete `Challenge`s (managed through `ChallengeStatus`), earn `Reward`s (handled through `PlayerReward`), and climb to the top of the `Leaderboard`s.

`Challenge`s describe the learning path for the game players, each rooting its own tree and wrapping one or more activities to be solved, possibly, with additional conditioners (e.g., using `mode` and `mode_parameters` to add time

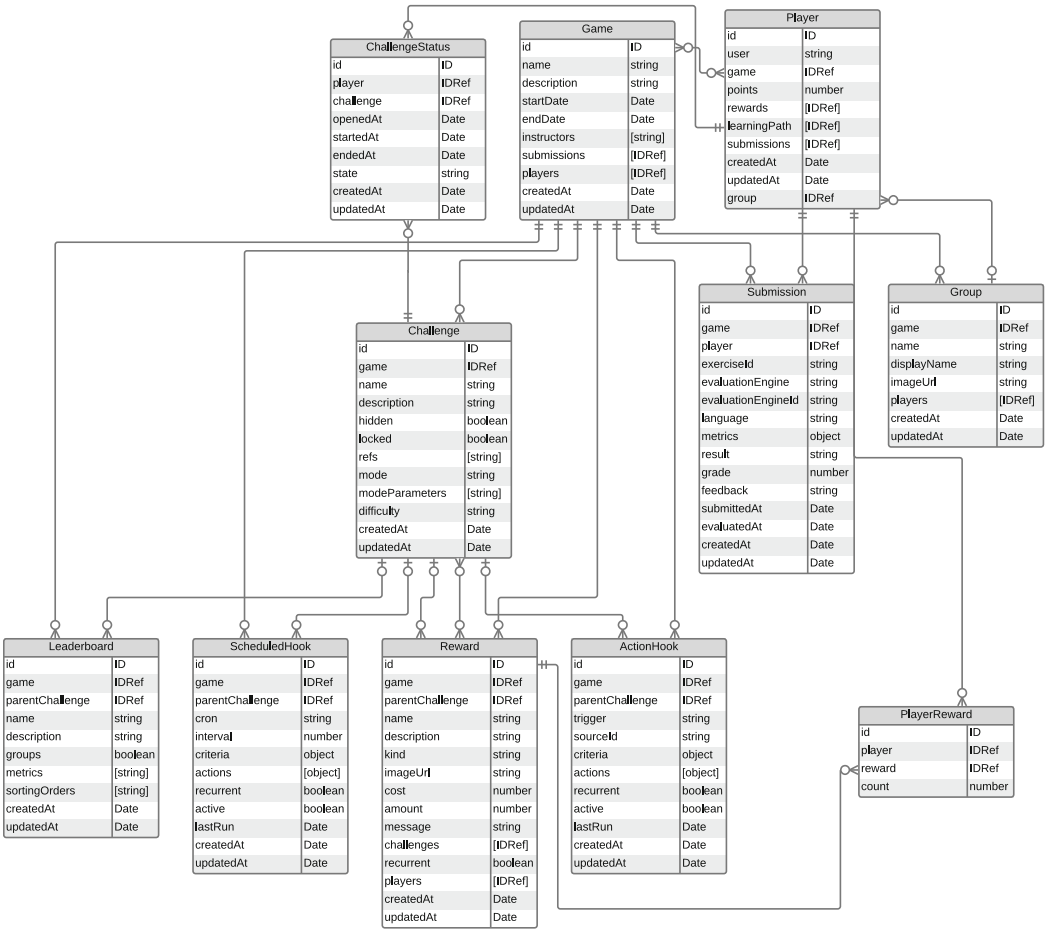

Fig. 4. Logical data model of FGPE GS.

constraints). Once a player enrolls in the game, its personalized learning path is generated from the challenges, i.e., a `ChallengeStatus` is created for each pair challenge-player with the state either `LOCKED` - if property locked of the `Challenge` is enabled -, `HIDDEN` - if property hidden of the `Challenge` is enabled -, or `AVAILABLE`.

`Player`s progress in the game by submitting attempts to solve activities, named `Submission`s. A `Submission` contains header information, such as the ID of the exercise – `exerciseId` – and the date of submission – `submittedAt` –, as well as data obtained from the evaluation, including the engine responsible for the evaluation – `evaluationEngine` –, the (programming) `language` of the attempt, evaluation `metrics`, `grade`, `result`, and `feedback`. Solving all activities of a challenge according to its constraints will trigger an hook to mark the challenge as `COMPLETED` and set the `endedAt` property to the current date.

There are many other internal hooks installed for a game, such as to notify players when the game starts and ends, assign rewards appended to a challenge when a player completes it, among others. Moreover, hooks are responsible for implementing most game logic as each rule from GEdIL also maps into a hook. Two types of hooks are supported by the service: `ScheduledHook`, which fires based on time (`cron`job or `interval`), and `ActionHook`, which is triggered both

directly or indirectly by an action (`trigger` and `source` indicate what type of action and from which object, respectively). Both kinds of hooks support `criteria` to check whether the hook `actions` should or not execute (see [13] to explore allowed values for criteria and actions). For instance, to assign a reward when a player reaches 20 submissions to exercise X, a hook must be defined with `SUBMISSION_RECEIVED` trigger, X in the `sourceId`, `criteria` checking whether submission count for exercise X is greater or equal to 20, and an `action` of type `GIVE` and the reward ID as its a single parameter.

`Reward`s have a `name`, a `description`, and a `kind`. Depending on its kind, a `Reward` may also have an image to display in the user interface – `rewardUrl` – (e.g., badges and virtual items), an `amount` (e.g., points), a `message` (e.g., hints), or references to `challenges` (e.g., revealables and unlockables). Furthermore, the author may opt to allow players to buy certain rewards, by setting their `cost` (in points) to a value greater than 0, and to accumulate multiple equal rewards, by enabling `recurrent` property. To keep tracking of accumulated rewards, the `PlayerReward` model holds the rewards collected by each player along with the count of repeated rewards.

In addition to progressing in the learning path and obtaining rewards, players also compete for a higher position in `Leaderboard`s defined. Leaderboards have a `name`, a `description`, and the `metrics` to sort players by their respective sorting order – `sortingOrders`. Metrics are JSONPath selectors over an object containing the player and its last submissions to each activity (e.g., for programming activities, Listing 1.1 sorts players by program size). Since leaderboards can be somewhat intimidating for underperforming students, teachers may separate students into `Group`s and enable `groups` property in the `Leaderboard` to display only their positions within the group. Notwithstanding, groups may be used by the instructors only as an organizational unit without any particular impact on the leaderboards, providing a `name` and, possibly, an image to refer to a set of players.

Listing 1.1. Example JSONPath selector to sort players by program size

```
$.latestSubmissions.[?(@.result=='ACCEPT')].metrics.programSize
```

4 Validation

The validation of the FGPE GS consists of end-to-end tests on the main functionality of the service. In particular, they include the following use cases: (1) author (i.e., a user with role `AUTHOR`) imports a GEdIL archive; (2) author assigns an instructor to a game; (3) teacher creates two groups; (4) teacher adds four players into the game; (5) teacher auto-assigns four enrolled players to two groups; (6) challenge unlocked after student submits a correct solution to each exercise of the preceding challenge; (7) badge granted to students who submit five wrong answers; and (8) badge granted to student who completes last challenge.

The tests were developed using Jest, a JavaScript library for creating, running, and structuring tests for any project built using JavaScript or TypeScript.

The database in MongoDB and the Redis instance, to support the queues, run in a Docker setup alongside the test runner. A total of 10 runs has been performed on the test machine (a mid-range DELL XPS 15 9570) under the same conditions and their execution times registered separately for each use case. Table 1 presents the minimum, maximum, and average execution times in milliseconds from the 10 runs, for each use case. All tests were correct from the functional perspective.

Table 1. Execution times of the end-to-end tests performed

Use case	1	2	3	4	5	6	7	8
Min. (ms)	106.823	21.651	29.130	45.467	85.286	113.538	102.644	46.953
Avg. (ms)	111.709	22.966	30.846	48.722	89.127	118.803	106.538	49.342
Max. (ms)	120.175	24.529	34.742	56.413	95.108	127.288	110.764	52.549

From the results, it is possible to notice a clear bottleneck in tasks involving file uploads (e.g., importing GEdIL archive and submissions), as expected (these are very rare tasks). Nevertheless, execution times are still great, considering that the requested response graphs, for all responses, were either two or three levels deep.

5 Conclusion

Gamification seems an effective technique to promote human engagement, which is essential in the realm of education, particularly in courses demanding much practice, such as computer programming. However, the necessary effort to gamify a specific course is many times higher than teachers are eager to take. Having a service that consumes our set of gamification rules and applies them to any set of non-gamified activities, would significantly reduce this cost, allowing convenient reusing, adapting, and implementing gamification rules in different subjects and courses.

This paper introduces FGPE GS, an open-source GraphQL service that accepts a package – adhering to a dedicated open-source language to describe gamification layers: GEdIL – containing the gamification rules and implements them over the automatically assessable activities referenced by the challenges, providing students with a thoroughly gamified experience. Although designed for programming education, neither GEdIL nor FGPE GS has any restriction that precludes its use in other educational subjects.

The work described here will be continued in the future according to the plan of the Framework for Gamified Programming Education project [4]. The only missing artifact to complete the ecosystem is the Learning Environment (i.e., the user interface), to which we will devote most of our attention. Nevertheless, additional unit tests and features (e.g., a simple user interface for administration, duels, etc.) will also complement the FGPE GS.

Acknowledgment. The work described in this paper was done within the Framework for Gamified Programming Education project (2018-1-PL01-KA203-050803) supported by the European Union's Erasmus Plus programme.

References

1. Almeida, F., Duarte Santos, J., Augusto Monteiro, J.: The challenges and opportunities in the digitalization of companies in a post-covid-19 world. IEEE Engineering Management Review, **48**(3), 97–103 (2020)
2. Bosse, Y., Gerosa, M.A.: Why is programming so difficult to learn? patterns of difficulties related to programming learning mid-stage. SIGSOFT Softw. Eng. Notes **41**(6), 1–6 (2017)
3. de Queirós, R.A.P.: A survey on game backend services. In: de Queirós, R.A.P., Teixeira Pinto, M. (eds.) Gamification-Based E-Learning Strategies for Computer Programming Education, pp. 1–13. IGI Global (2017)
4. FGPE Consortium. Framework for Gamified Programming Education (2018). https://fgpe.usz.edu.pl. Accessed on 8 Dec 2020
5. GraphQL Foundation. GraphQL Specification (2019). https://spec.graphql.org. Accessed 6 Dec 2020
6. Indeed.com. What careers are most in-demand right now? (2020). https://www.indeed.com/career-advice/finding-a-job/in-demand-careers. Accessed 6 Dec 2020
7. Keycloak. Keycloak: open source identity and access management solution (2014). https://www.keycloak.org. Accessed 9 Dec 2020
8. Leal, J.P., Silva, F.: Mooshak: a web-based multi-site programming contest system. Software: Practice and Experience, **33**(6), 567–581 (2003)
9. Mysliwiec, K.: NestJS: A Progressive Node.js Framework (2018). https://nestjs.com. Accessed 6 Dec 2020
10. Paiva,, J.C., Leal, J.P., Queirós, R.: Odin: a service for gamification of learning activities. In: Sierra-Rodríguez, J.L., Leal, J.P., Simões, A. (eds.) Languages, Applications and Technologies - 4th International Symposium, SLATE 2015, Madrid, Spain, June 18-19, 2015, Revised Selected Papers, vol. 563, Communications in Computer and Information Science, pp. 194–204. Springer (2015)
11. Prenkaj, B., Stilo, G., Madeddu, L.: Challenges and solutions to the student dropout prediction problem in online courses. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM 2020, pp. 3513–3514, New York, NY, USA (2020). Association for Computing Machinery
12. Rojas-López, A., Rincón-Flores, E.G., Mena, J., García-Peñalvo, F.J., Ramírez-Montoya, M.S.: Engagement in the course of programming in higher education through the use of gamification. Universal Access Inf. Soc. **18**(3), 583–597 (2019)
13. Swacha, J., Paiva, J.C., Leal, J.P., Queirós, R., Montella, R., Kosta, S.: GEdIL - gamified education interoperability language. Information **11**(6), 287 (2020)