information

Article

PROud—A Gamification Framework Based on Programming Exercises Usage Data

Ricardo Queirós

Department of Informatics, School of Media Arts and Design (ESMAD), Polytechnic of Porto, 4480-786 Vila do Conde, Portugal; ricardoqueiros@esmad.ipp.pt; Tel.: +351-966531586

Received: 8 January 2019; Accepted: 5 February 2019; Published: 7 February 2019

Abstract: Solving programming exercises is the best way to promote practice in computer programming courses and, hence, to learn a programming language. Meanwhile, programming courses continue to have an high rate of failures and dropouts. The main reasons are related with the inherent domain complexity, the teaching methodologies, and the absence of automatic systems with features such as intelligent authoring, profile-based exercise sequencing, content adaptation, and automatic evaluation on the student's resolution. At the same time, gamification is being used as an approach to engage learners' motivations. Despite its success, its implementation is still complex and based on ad-hoc and proprietary solutions. This paper presents PROud as a framework to inject gamification features in computer programming learning environments based on the usage data from programming exercises. This data can be divided into two categories: generic data produced by the learning environment—such as, the number of attempts and the duration that the students took to solve a specific exercise—or code-specific data produced by the assessment tool—such as, code size, use memory, or keyword detection. The data is gathered in cloud storage and can be consumed by the learning environment through the use of a client library that communicates with the server through an established Application Programming Interface (API). With the fetched data, the learning environment can generate new gamification assets (e.g., leaderboards, quests, levels) or enrich content adaptations and recommendations in the inner components such as the sequencing tools. The framework is evaluated on its usefulness in the creation of a gamification asset to present dynamic statistics on specific exercises.

Keywords: cloud gamification; web services; computer programming

1. Introduction

Nowadays, computer programming proves to be a crucial domain not only in university education, but also in younger education. This trend is related to the fact that programming is considered one of the best ways to stimulate articulated and logical reasoning in problem solving, which we have to constantly face in our daily lives. Despite its importance, teaching in programming courses continues to be problematic due to several issues: (1) it is a complex domain that requires the student to master a panoply of concepts; (2) classroom methodologies are still composed of theoretical presentations of language syntax that do not enhance programming practice, which is crucial for learning in this domain; (3) courses have an extensive curriculum, typically with large classes, which makes it difficult for teachers to give rich and consistent feedback to all students.

Learning computer programming can be a lonely, complex, and demotivating process [1–3]. These issues have been addressed in the last years, with the appearance of several online learning environments trying to leverage coding education and make it accessible to everyone, even those with absolutely no coding experience or knowledge [4,5]. These environments come in various formats ranging from non-interactive approaches (e.g., YouTube channels, blogs, books) to integrated and interactive solutions (e.g., intelligent tutors, online coding providers, Massive Open Online Courses (MOOCs)) [6].

In order to overcome these issues, several systems have emerged in recent years to automate the teaching–learning process of computer programming [7]. These systems were given various names: interactive learning environments, online playgrounds, learning management systems, MOOCs, and intelligent tutors among others. Regardless of their names, their main goal is to make available interactive 24x7 environments where students have at their disposal a set of programming exercises to solve. After their resolution and submission, students receive automatic feedback sent through an online assessment system. With this approach, the teacher relieves their manual evaluation work and focuses on other important aspects of the teaching process, such as the creation of new programming exercises. Despite their partial success, these systems present several problems related to evaluation, interoperability, user experience, and adaptability among others [8]. Regarding evaluation, existing systems are limited to give concise feedback on students' performances; they do not go into details about the code, such as the use of certain keywords, the code style used, or the implemented algorithm. These systems also have poor interoperability features, for instance, not allowing a flexible integration with the existing learning management systems. The user experience is often neglected causing latencies in page loading or remote requests. This is often due to the fact that online environments try to mimic traditional integrated development environments, making interaction more complex and time consuming. In terms of adaptability, most systems present a rigid sequence of exercises to be solved, with unsuitable content sequencing and a great inability to adapt to the learner's profile.

One of the great trends of the last decade is the use of gamification in several types of environments [9,10], including online programming learning environments [11]. The goal is simple: keep students motivated and encourage more practice in order to gain programming skills. Although it is a legitimate goal, the gamification used is limited to a set of badges that are earned when the student completes a particular module or a set of rankings, to compare the students' performance in the course. Despite being a step forward in the gamification of these environments, there is still much more to explore in order to stimulate the students' competitiveness [12]. In order to do this, our proposal is to gather information about the usage data of the programming exercises. Such information can then be stored in a central storage and, through a programming interface, can be used in a flexible way, as input to various components of the learning system or in the construction of specific gamification assets (e.g., leaderboards, achievements, statistical panels).

This paper presents PROud—a framework to foster gamification features in learning environments based on the usage data of programming exercises. The framework is composed of several components: a storage where all usage data is stored; an API to expose filtered usage data; and a client library that will allow any web developers to rapidly inject multiple gamification features and integrate them seamlessly into a computer programming teaching–learning environment. In order to evaluate PROud, we used it for the construction of a gamification asset to present programming exercises statistics.

The article is structured as follows: Section 2 presents the various tools that typically arise in a teaching–learning computer programming ecosystem. Afterwards, PROud is presented as a framework to facilitate the creation of gamification elements based on the the usage data from programming exercises. Then, one case study is presented where the framework can be used, more precisely, in the creation of a statistics panel related to a given programming exercise. Finally, conclusions are presented regarding this work and a set of improvements to be made in the short term are discussed.

2. Programming Exercises Learning Environment Ecosystem

Nowadays, programming exercises learning environments (PELEs) are typically composed by a set of independent tools dispersed in the cloud, which are responsible to execute specialized tasks. However, these tools cannot afford to be isolated from the educational institution ecosystem. Thus, the potential for interoperability is an important, although frequently overlooked, aspect in this domain [13].

In the past, we had monolithic environments where all the features were implemented in the same system and on the same machine [14]. Then, with the advent of service-oriented architectures (SOAs), developers began to create smaller client apps getting most of their features through well established service APIs. However, in the PELE context, this approach did not work so well, since there is a lack of integration standards [15]. Apart from these issues, some environments present common tools such as:

- A learning management system—a learner's entry point to launch the PELE,
- An assessment tool—to evaluate a learner's attempt to solve a specific exercise,
- A programming exercise repository—to store and make exercises discoverable, and
- A resource sequencing tool—to present and organize exercises in an intelligent way.

Meanwhile, other services appeared, to enrich PELE ecosystems, as secondary services. Figure 1 depicts a typical PELE architecture.

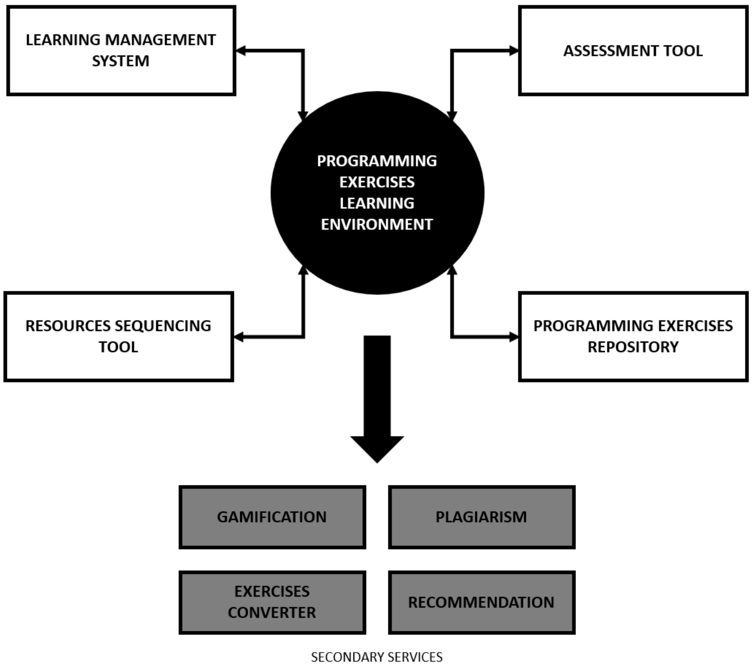

LEARNING MANAGEMENT SYSTEM
ASSESSMENT TOOL
PROGRAMMING EXERCISES LEARNING ENVIRONMENT
RESOURCES SEQUENCING TOOL
PROGRAMMING EXERCISES REPOSITORY
GAMIFICATION
PLAGIARISM
EXERCISES CONVERTER
RECOMMENDATION
SECONDARY SERVICES

Figure 1. Programming exercises learning environment ecosystem.

In the next subsections, we detail these common tools and the major initiatives of interoperability for each tool mentioned.

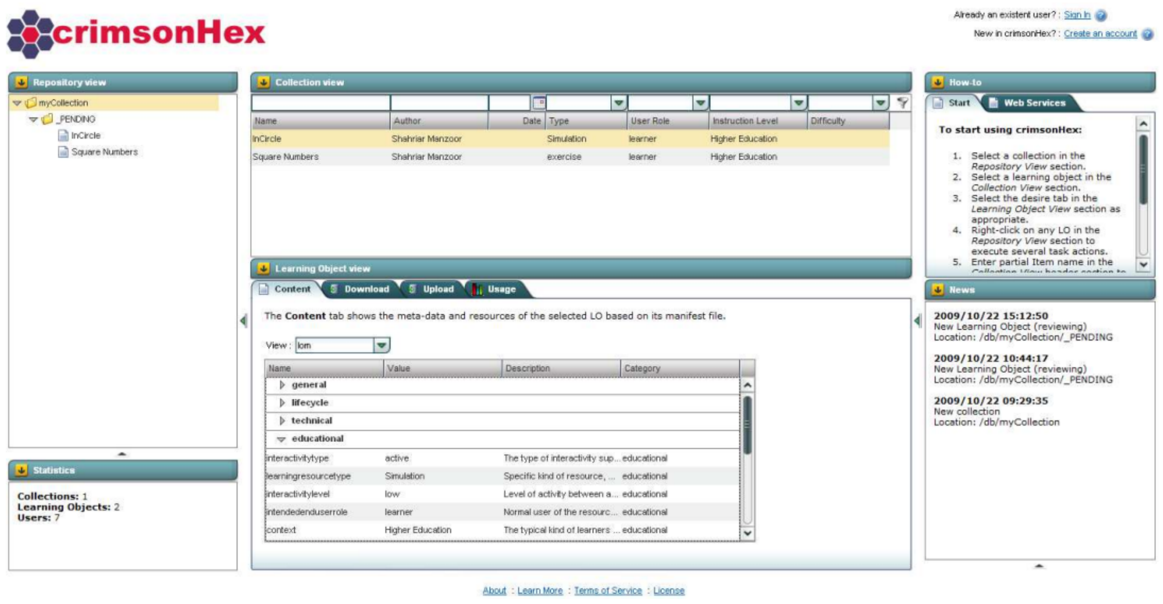
2.1. Learning Management Systems

A learning management system (LMS) is a software application for the administration, documentation, tracking, reporting, and delivery of educational courses, training programs, or learning and development programs. The LMS market is expected to be worth over $15.72 billion by 2021 [16]. Not only universities, but companies now also use some form of educational technology to instruct employees during formal learning hours. In the educational realm, the LMS is used by two user groups: learners and teachers. The learners can use the LMS to progress their learning experience and to collaborate with their colleagues; the teachers can deliver educational resources and track, analyze, and report on the learners' evolution.

There were several approaches to integrate LMS with other systems, including defining LMS from scratch based on service-oriented architectures (SOA) [13], integrating web services layers within the LMS infrastructure [15], and providing support for interoperability specifications [17]. The latter approach is mainly based on IMS Global learning Consortium specifications, namely the LTI (learning tools interoperability) specification that facilitates the integration between LMS and external applications. The main goal of the LTI is to standardize the process for building links between learning tools and the LMS.

2.2. Programming Exercise Repositories

A learning objects repository (LOR) is a system that stores educational resources and enables educators to manage, share, and use them. These resources (or learning objects) are small, self-contained, and reusable educational units that, typically, have additional metadata to facilitate their cataloging and discovery in searches. There are several programming exercise repositories (e.g., Universidad de Valladolid (UVA) online repository and CloudCoder), but none of them implements any interoperability standard. One notable exception is the crimsonHex (Figure 2) [18]—a service-oriented repository of learning objects. This repository supports new definitions of programming exercises as learning objects. The repository is also fully compliant with existing communication standards, namely, the IMS Digital Repositories Interoperability (DRI). The purpose of this specification is to provide recommendations for the interoperation of the most common repository functions. These recommendations should be implementable across services to enable them to present a common interface.



Name	Author	Date	Type	User Role	Instruction Level	Difficulty
InCircle	Shahriar Manzoor		Simulation	learner	Higher Education	
Square Numbers	Shahriar Manzoor		exercise	learner	Higher Education	

The **Content** tab shows the meta-data and resources of the selected LO based on its manifest file.

Name	Value	Description	Category
general			
lifecycle			
technical			
educational			
interactivitytype	active	The type of interactivity sup...	educational
learningresourcetype	Simulation	Specific kind of resource, ...	educational
interactivitylevel	low	Level of activity between a...	educational
intendedenduserrole	learner	Normal user of the resourc...	educational
context	Higher Education	The typical kind of learners ...	educational

Collections: 1
Learning Objects: 2
Users: 7

To start using crimsonHex:

1. Select a collection in the *Repository View* section.
2. Select a learning object in the *Collection View* section.
3. Select the desire tab in the *Learning Object View* section as appropriate.
4. Right-click on any LO in the *Repository View* section to execute several task actions.
5. Enter partial Item name in the

2009/10/22 15:12:50
New Learning Object (reviewing)
Location: /db/myCollection/_PENDING

2009/10/22 10:44:17
New Learning Object (reviewing)
Location: /db/myCollection/_PENDING

2009/10/22 09:29:35
New collection
Location: /db/myCollection

Figure 2. The crimsonHex programming exercise repository.

The crimsonHex repository also extends the DRI specifications with new functions, formalizing message interchange and providing a Representational State Transfer (REST) interface.

2.3. Assessment Tools

The ultimate goal of an assessment tool is to mark and grade programming exercises. These types of tools can perform two types of code analysis: static or dynamic.

Static code analysis is the process of detecting errors and defects in a software's source code. Static analysis can be viewed as an automated code review process. Dynamic code analysis is the method of analyzing an application during its execution. The dynamic analysis process can be divided into several steps: preparing input data, running a test program launch, gathering the necessary parameters, and analyzing the output data.

Most of the assessment tools (e.g., Mooshak [19] and DOMJudge) implement the second type of code analysis. Typically, these types of tools perform four tasks:

- They receive references from the exercise and learner and an attempt to solve the exercise (typically a program);
- They load the exercise from the repository using the given references;
- They compile the solution and run a set of tests, related to the exercise, against the attempt of the learner; and
- They produce an evaluation report with the classification and feedback.

Most of the assessment tools run standalone or can be accessed online through a simple Graphical user Interface (GUI). To the best of the author's knowledge, there is no programming exercise evaluation service [20]. The only active approach is the Evaluate Programming Exercise [21]. This service models the evaluation of an attempt to solve a programming exercise defined as a learning object and produces a detailed report. This evaluation report includes information to support exercise assessment and grading by client systems. The three types of requests handled by this service are:

- ListCapabilities—providing the client systems with evaluator capabilities;
- EvaluateSubmission—allowing the request for a programming exercise evaluation; and
- GetReport—allowing a requester to get an evaluation report using a ticket.

2.4. Resource Sequencing Tool

A resource sequencing tool aims to provide a logical sequence of resources to the learner. Typically, the exercises are sorted from the easiest exercises to the most difficult ones and organized into different modules. The main idea behind the fragmentation in modules is so that, at the end of a module, a learner can receive a notification on how they performed on the central theme of this module, enabling them to jump to the next challenge. Some systems use gamification techniques with module blocking; some modules are only available after the learner has resolved all exercises (or a percentage) of the previous module.

One interesting work is Seqins, which can be defined as a sequencing tool of digital educational resources [22]. Seqins includes a flexible sequencing model that fosters students to learn at different rhythms. Seqins also support precedence among content units, assessment results, and students' progress, providing an XML representation of the resources to present to the current learner.

3. PROud Framework

This paper presents PROud (which stands for PROgramming Usage Data) as a simple gamification framework based on the usage data from programming exercises. The main goal of the framework is to facilitate the development of gamification elements such as leaderboards, achievements, and statistics, as well as to enrich others such as resource sequencing components.

3.1. Architecture

The architecture of the framework can be seen in Figure 3.

The framework is based on a client–server model:

- Client—a client web application (PELE) with the PROud library.
- Server—a container installed on the server with the PROud engine.

On the client side, the developer has to download the PROud client library (encoded in JavaScript) and reference it in the learning environment (PELE) code. The library can be used by two types of components: (1) Producers: responsible for generating the usage data. Typically producers will be Integrated Development Environments (IDEs) and evaluation tools. The library receives this data and sends it to the server; and (2) Consumers: responsible for using the library to obtain server-specific usage data, in order to create new gamification elements or enrich existing components.

On the server side, the PROud engine is installed as a docker container on a server machine. The engine's mission is to receive query and mutation requests, resolve them by interacting with the associated cloud storage, and respond, with the filtered data, back to the client.

In the next sections we detail both aspects of the framework, namely its components and the interactions between them.

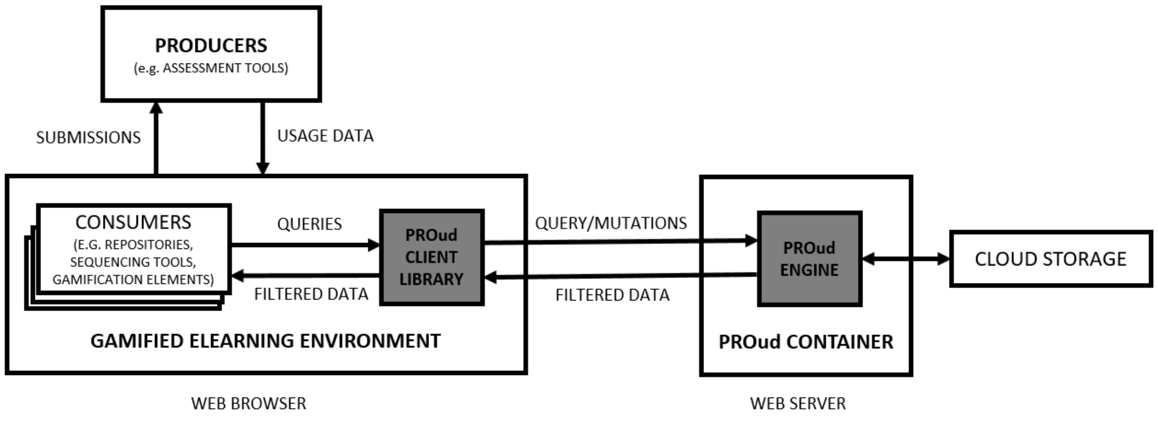


Figure 3. PROud architecture.

3.2. PROud Engine

The PROud engine is the main server component of the framework. It can be installed as a docker container, which is a software technology that provides independent containers to run within a single Windows or Linux instance, avoiding the overhead of starting and maintaining virtual machines (VMs).

The server architecture is illustrated in Figure 4:

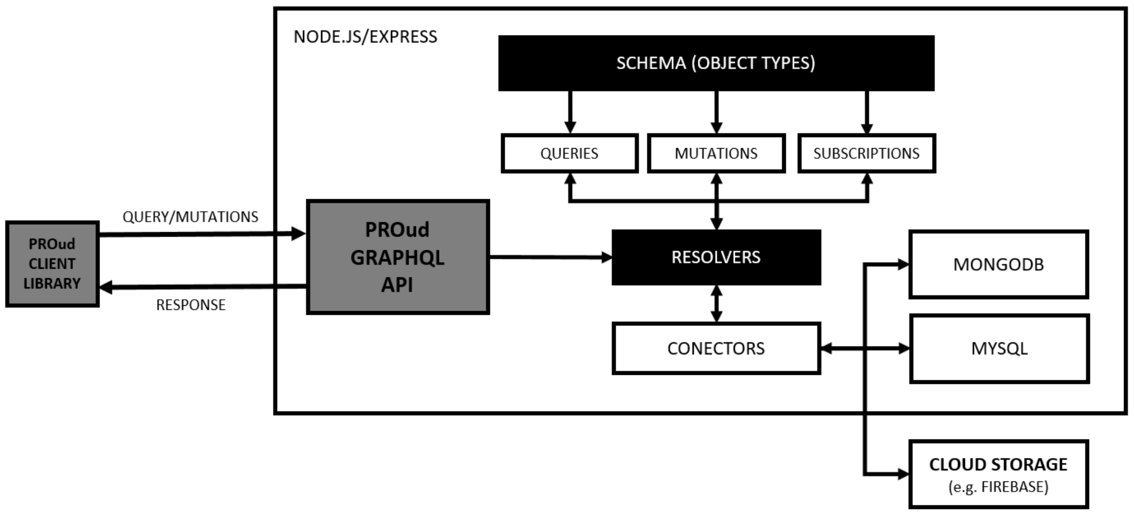


Figure 4. PROud server engine.

The engine runs on an instance of a GraphQL server which includes a GraphQL API (http://graphql.org/). This is an alternative to the REST software architectural style, allowing developers to provide a complete and understandable description of the API data. Furthermore, it gives clients the power to ask for exactly what they need and nothing more.

The simplest way to run a GraphQL API server is to use Express, a popular web application framework for Node.js. The following code excerpt shows the initial code setup in order to run a GraphQL server:

Listing 1: Server initial setup

```
const express = require('express');
const graphqlHTTP = require('express-graphql');
const { buildSchema } = require('graphql');

// PROud schema using GraphQL schema language
const schema = buildSchema(`
... schema ...
`);

// Root provides a resolver function for each API endpoint
const root = {
firstEndPoint: () => { return ...; }
};

// use the express-graphql library to mount a
// GraphQL API server on the /graphql HTTP endpoint
const app = express();
app.use('/graphql', graphqlHTTP({
schema: schema,
rootValue: root,
graphiql: true,
}));
app.listen(4000);
console.log('Running a GraphQL API server at localhost:4000/graphql');
```

3.2.1. Data Model

Using the PROud framework developers will have the chance to gather all the usage data on a specific programming exercise. The usage data will be grouped by programming exercise and organized in two categories:

- Generic usage data—metadata (not tied to any specific domain) related to the solving activity.
- Code usage data—usage data mainly related to code performance metrics.

Generic usage data is all the information linked to a specific submission regardless of the domain. Typically, we will have data on initial/end date of the solving activity, the number of attempts, the final submission success, and the difficulty level (defined by the learner at the end of the solving activity).

Regarding code usage data, there are several performance metrics which can be calculated by producer tools (e.g., assessment tools), such as, execution time, compilation time, code size, memory space, power consumption, and other computing resources.

Table 1 summarizes the most important data types for both flavors of data:

Table 1. PROud data model.

Resources	Generic Usage Data	Code Usage Data
ID	Start date	Code_size
Title	End date	Use_memory
Description	Attempts	Execution_time
Image	Success	Test_creation
Author	Difficulty_level	Have_comments
Link		

In order to formalize the previous data model, we will use the GraphQL type system. GraphQL uses a strong type system to define the features of an API. All types exposed by the API are formalized in a schema using the GraphQL schema definition language (SDL), which can be defined as a contract between the client and the server to define how a client can access the data. Once the schema is defined, developers (both from the frontend and backend) can start working without further communication since both are aware of the final data structure that is sent over the network.

Using the SDL, all one needs to do is specify the types for their API and inject them as an argument in the buildSchema function. The following code excerpt presents the schema for the programming exercise usage data domain:

Listing 2: PROud schema language

```
// Construct a schema
const schema = buildSchema('
type Resource {
resourceId: ID!
title: String!
description: String!
image: String!
author: String!
link: String!
submissions: [Submission!]!
}
type Submission {
submissionId: ID!
userId: String!
startDate: String!
endDate: String!
attempts: Int!
success: Boolean!
difficultyLevel: Level!
codeMetrics: [CodeMetrics!]!
}
enum Level {
BEGINNER
EASY
NORMAL
HARD
EXPERT
}
type CodeMetrics {
codeSize: Int!
executionTime: Int!
useMemory: Int!
testsCreation: Boolean!
haveComments: Boolean!
}
');
```

A resource (programming exercise) is the main unit of the data model. For each resource there are several submissions. A submission is made by a learner. It has a start and end date. During this period the learner can submit several attempts. The last attempt is evaluated by the assessment tool and registered in the success field. The learner also has the chance, in the end of the solving activity, to grade the exercise based on the difficulty level. Finally, the code metrics are associated to the last attempt.

3.2.2. Queries and Mutations

Most types in a GraphQL schema will just be normal object types, but there are two types that are special within a schema: queries and mutations. The former are used by the client to request the data it needs from the server. The later, are used to CUD (create, update, delete) the data; more precisely, to create new data, Update existing data, and delete existing data.

Every GraphQL service has a query type and may or may not have a mutation type. These types are the same as a regular object type, but they are special because they define the entry point of every GraphQL query.

The following code shows two queries:

Listing 3: Query example

```
// Two queries
type Query {
resources: [Resource!]!
resource(id: ID!): Resource
}
...
// Resolver function for each API endpoint
const root = {
resources: () => proud,
resource: (args) => proud.filter(resource => resource.id === args.id)[0]
}
```

The first query retrieves all the resources from the cloud storage (for legibility reasons: proud is a hardcoded as an array of resources). The second query returns a specific resource (given its ID) from the cloud storage.

Next, we present a mutation to add a new submission for a specific resource given its ID:

Listing 4: Mutation example

```
type Mutation {
createSubmission(resId: Int, input: Submission)
}

const root = {...}
createSubmission: function ({id, input}) {
// Create a new submission for a given resource
const resource = proud.filter(resource => resource.id === id)[0]
resource.submissions.push(input)
}
}
```

The resolvers will run the associated code and get/put data from/to the cloud storage. The interaction between both is not direct. In this case, we use connectors as mediators in order to abstract the data layer. This way, we can plug any storage type into the framework.

3.3. Client Library

The client library will allow the PELE to access the server in a simple way. The library will be accessed by producer tools, to send programming exercise usage data, and by consumer tools, to obtain usage data and build new gamification elements or enrich other components.

Another gain of the library is in abstracting the use of GraphQL. Through a set of simple methods, the library has a reduced learning curve allowing anyone to interact with the library at a fast pace.

Table 2 shows the most important methods that developers should use to interact with the PROud library.

Table 2. PROud client library main methods.

Methods	**Description**
createSession()	Creates a session for the exercises solving activity
submitData(resId, userId, submission)	Produces a user's final submission for a specific exercise
getData(resId)	Gets information on a specific resource
getData(GraphQL)	Gets information on a specific resource based on a GraphQL query
loadTemplate(strTemplate)	Loads a Cascading Style Sheet (CSS) template for the GUI component
addTab(index, caption)	Adds a tab for the GUI component
addField(location, caption, value, [type])	Adds a field in a specific tab

In its core, the library uses the GraphQL request client (https://github.com/prisma/graphql-request). The next code excerpt shows part of the library method's internal code:

Listing 5: Internal library code

```
import { GraphQLClient } from 'graphql-request'

async function getData(resId) {
const endpoint = 'SERVER_URI'
const query = /* GraphQL */ `
{
Resource(resourceId: resId) {
title,
description,
...
submissions {
submissionId,
userId,
startDate,
...
}
}
}
const graphQLClient = new GraphQLClient(endpoint, {
credentials: 'include',
mode: 'cors',
})
const data = await graphQLClient.request(query)
return JSON.stringify(data, undefined, 2)
}
// Other functions
...
```

4. Example of Application

The framework was evaluated on its usefulness in one case study: the creation of a gamification asset presenting dynamic statistics on specific exercises.

One of the main goals of the framework is to help developers to create gamification assets. In this section, we present the creation of a statistics panel in a PELE using the PROud client library. The main goals of the library are to abstract server data access using utility functions and to produce sophisticated GUI components based on predefined templates.

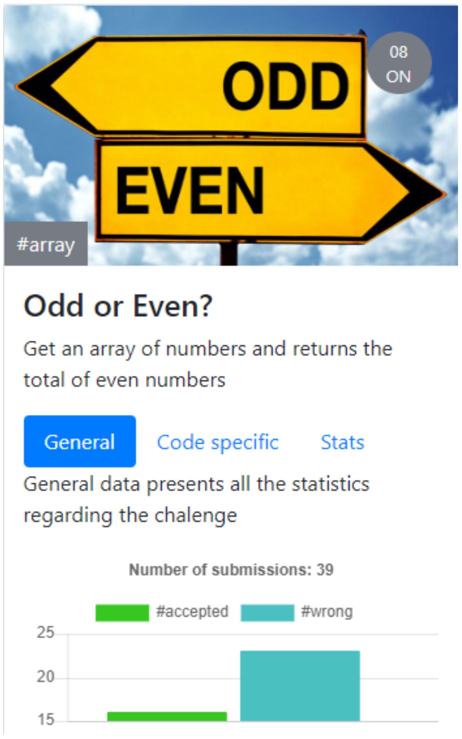
For the creation of this gamification asset, developers must get the resource data from the server and then, load a specific template to act as a GUI container. Inside the container, developers should use the addTab method of the PROud library to organize the data in different tab panes. For each tab pane, developers should use addField method for effectively inject the server data in the GUI component. The method has several signatures (overloading). Typically, we must define where the data will appear inside the component, then define a caption and, finally, inject the data to be rendered. By default, the field is formatted as text. If you want to change the field type to a chart, you must define the fields to appear in the chart and the chart type.

The next code excerpt presents the main code for the creation of a simple resource statistics panel. First, it imports the library in the PELE code. Then, the developer needs to create the following function using the library:

Listing 6: Creation of a gamification asset—statistics panel

```
...
// Create a new Proud instance based on a sessionId
Proud p = new Proud(sessionId)
// Get data from the server based on a specific resourceId
ProudResource r = p.getData(resourceId)
...
function createPanel() {
ProudTemplate template = p.loadTemplate("card_stats.tpl")
template.addTab(0,"General")
template.addTab(1,"Code Specific")
template.addTab(2,"Stats")
// Fill the placeholders of the template with server filtered data
template.tabs[0].addField(TEMPLATE.HEADER, "", r.imagePath)
template.tabs[0].addField(TEMPLATE.TITLE, "", r.title)
...
template.tabs[0].addField(TEMPLATE.BODY,
"Number of submissions",
r.submissions.length)
// Calculate new fields based on server data
ProudField pf1 = new ProudField("#accepted", r.submissions.success===true)
ProudField pf2 = new ProudField("#wrong", r.submissions.success===false)
// Aggregate fields and change their default type (TEXT)
template.tabs[0].addField(TEMPLATE.BODY,
"",
[pf1,pf2],
TEMPLATE.BODY.BARCHART)
...
}
...
}
```

The above code will produce the following asset (Figure 5) in the PELE GUI.



Odd or Even?

Get an array of numbers and returns the total of even numbers

General | Code specific | Stats

General data presents all the statistics regarding the chalenge

Figure 5. Statistics panel.

It is important to note that the library can be used by the data producers to submit data to the server. There are (typically) two data producers: the learning environment for generic usage data and the assessment tool for code usage data. Both data types are gathered in a *submission* object and submit to the server using the client library function $submitData(resId, userId, submission)$. The following code shows how the learning environment uses this function:

Listing 7: Submission of data to the server

```
...
Proud p = new Proud(sessionId)
const submission = {
startDate:"Thu Jan 31 2019 18:03:36",
endDate:"Thu Jan 31 2019 18:23:16",
attempts:2,
success:true,
codeMetrics:[{codeSize:12, executionTime:11, useMemory:23},...]
}
p.submitData(resourceId, userId, submission)
...
}
```

Regarding code metrics, this kind of data is often generated by the assessment tool. It is up to the developer gather that information and inject it into the *submission* object.

5. Conclusions

Cloud gamification is now entering at full power in online learning environments. Regarding the computer programming domain, there is a large margin of progression in the use of gamification. The main goals of this approach are clear: to increase students' motivation and to enhance their competitive spirit. However, despite the main advantages of this approach there are a number of issues that have not yet been addressed, such as the lack of specific services in this area and the poor interoperability of existing services.

The PROud framework tries to fill this gap by providing a very simple library which can be used to access an API to query and mutate the usage data from programming exercises. With this data, developers can create sophisticated GUI components, based on CSS templates, such as leaderboards, achievements, and statistics. The data can even serve as input for other components to adapt the content based on specific data. The templates architecture allows a web designer to enrich the framework with new templates.

In order to evaluate its usefulness, we implemented the library to create a programming exercise statistics panel. This test proves that it is very easy to use, abstracting the developer to deal with GraphQL details.

In future work, we will improve the template architecture and create documentation of the PROud framework on GitHub.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ala-Mutka, K.M. A Survey of Automated Assessment Approaches for Programming Assignments. *Comput. Sci. Educ.* **2005**, *15*, 83–102. [CrossRef]
2. O'Kelly, J.; Gibson, J.P. RoboCode & Problem-based Learning: A Non-prescriptive Approach to Teaching Programming. *SIGCSE Bull.* **2006**, *38*, 217–221.
3. Robins, A.; Rountree, J.; Rountree, N. Learning and Teaching Programming: A Review and Discussion. *Comput. Sci. Educ.* **2003**, *13*, 137–172. [CrossRef]
4. Verdú, E.; Regueras, L.M.; Verdú, M.J.; Leal, J.P.; de Castro, J.P.; Queirós, R. A Distributed System for Learning Programming On-line. *Comput. Educ.* **2012**, *58*, 1–10. [CrossRef]
5. Coelho, A.; Kato, E.; Xavier, J.A.; Gonçalves, R. Serious Game for Introductory Programming. In Proceedings of the Second International Conference on Serious Games Development and Applications (SGDA'11), Lisbon, Portugal, 19–20 September 2011; pp. 61–71.
6. De Queirós, R.A.P. *Code Generation, Analysis Tools, and Testing for Quality*; IGI-Global: Hershey, PA, USA, 2019.
7. De Queirós, R.A.P. A Survey on Computer Programming Learning Environments. In *Code Generation, Analysis Tools, and Testing for Quality*; IGI-Global: Hershey, PA, USA, 2019; Chapter 4, pp. 90–105.
8. Von Hausswolff, K. Hands-on in Computer Programming Education. In Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17), Tacoma, WA, USA, 18–20 August 2017; pp. 279–280.
9. Ortiz, M.; Chiluiza, K.; Valcke, M. Gamification in Computer Programming: Effects on learning, engagement, self-efficacy and intrinsic motivation. In Proceedings of the European Conference on Game Based Learning, Graz, Austria, 5–6 October 2017.
10. Swacha, J.; Muszynska, K. Design patterns for gamification of work. In Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality, Salamanca, Spain, 2–4 November 2016, pp. 763–769.
11. Ahn, J.; Butler, B.S.; Alam, A.; Webster, S.A. Learner participation and engagement in open online courses: Insights from the Peer 2 Peer University. *MERLOT J. Online Learn. Teach.* **2013**, *9*, 160–171.
12. Layth Khaleel, F.; Ashaari, N.; Wook, T.; Tengku Wook, T.S.M.T.W.; Ismail, A. The study of gamification application architecture for programming language course. In Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication, Bali, Indonesia, 8–10 January 2015.
13. Queirós, R.; Leal, J.P.; Paiva, J.C. Integrating rich learning applications in LMS. In *State-of-the-Art and Future Directions of Smart Learning*; Springer: New York, NY, USA, 2016; pp. 381–386.

14. Dagger, D.; O'Connor, A.; Lawless, S.; Walsh, E.; Wade, V.P. Service-oriented e-learning platforms: From monolithic systems to flexible services. *IEEE Internet Comput.* **2007**, *11*, 28–35. [CrossRef]
15. Leal, J.P.; Queirós, R. Using the Learning Tools Interoperability Framework for LMS Integration in Service Oriented Architectures 2011. Available online: http://recipp.ipp.pt/handle/10400.22/4724 (accessed on 6 February 2019).
16. LMS Market by Component (Solution and Services), Delivery Mode (Distance Learning, Instructor-Led Training and Blended Learning), Deployment Type, User Type (Academic and Corporate), and Region - Global Forecast to 2023. Technical Report, 2018. Available online: https://www.marketsandmarkets.com/Market-Reports/learning-management-systems-market-1266.html (accessed on 6 February 2019).
17. Leal, J.P.; Queirós, R. A comparative study on LMS interoperability. In *Higher Education Institutions and Learning Management Systems: Adoption and Standardization*; IGI Global: Hershey, PA, USA, 2012.
18. Leal, J.; Queirós, R. CrimsonHex: A Service Oriented Repository of Specialised Learning Objects. In *Enterprise Information Systems*; Filipe, J., Cordeiro, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 24, pp. 102–113.
19. Leal, J.P.; Silva, F. Mooshak: A Web-based multi-site programming contest system. *Softw. Pract. Exp.* **2003**, *33*, 567–581. [CrossRef]
20. Al-Smadi, M.; Gütl, C. SOA-based Architecture for a Generic and Flexible E-assessment System. In Proceedings of the IEEE Conference on Education Engineering (EDUCON), Madrid, Spain, 14 April 2010; pp. 493–500.
21. Queirós, R.; Leal, J.P. Orchestration of E-Learning Services for Automatic Evaluation of Programming Exercises. *J. Univers. Comput. Sci.* **2012**, *18*, 1454–1482.
22. Queirós, R.; Leal, J.P.; Campos, J. Sequencing educational resources with Seqins. *Comput. Sci. Inf. Syst.* **2014**, *11*, 1479–1497. [CrossRef]