Odin: A Service for Gamification of Learning Activities

José Paulo Leal[1], José Paiva[2], and Ricardo Queirós[3]

[1] CRACS & INESC-Porto LA, Faculty of Sciences,
University of Porto, Porto, Portugal `zp@dcc.fc.up.pt`
[2] CRACS & INESC-Porto LA, Faculty of Sciences,
University of Porto, Porto, Portugal `up201200272@alunos.dcc.fc.up.pt`
[3] CRACS & INESC-Porto LA & DI/ESEIG/IPP,
Porto, Portugal `ricardoqueiros@eseig.ipp.pt`

Abstract. Existing gamification services have features that preclude their use by e-learning tools. Odin is a gamification service that mimics the API of state-of-the-art services without these limitations. This paper describes Odin, its role in an e-learning system architecture requiring gamification, and details its implementation. The validation of Odin involved the creation of a small e-learning game, integrated in a Learning Management System (LMS) using the Learning Tools Interoperability (LTI) specification.

Keywords: Gamification, E-Learning, Game services, Interoperability

1 Introduction

The use of game concepts and mechanics in non-game contexts is an effective way to engage users. Gamification is currently a word of order in different domains, from marketing to e-learning [2]. The massive use of this approach led to the concept of gamification as a service, provided by major players such as Google and Microsoft. These services leverage on their large user base to provide support for game progress mechanics such as points, leaderboards and badges, without requiring a specific authentication from the client application.

Gamification services are a great advantage to small web and tablet based applications, in particular to games. The game progress mechanics features provided by these services are also relevant in e-learning. However, e-learning systems are typically deployed in environments with a single sign-on managed by an academic institution. It would be unacceptable to require students to have an account with a third party such as Google, for instance.

The purpose of the Odin service is to provide a gamification service similar to the state of the art, without requiring registration of the end users. Its API is inspired in the Google Play Game Service (GPGS) with minor adjustments regarding user identification.

The remainder of this paper is organised as follows. Section 2 reviews the state of the art in game services. Section 3 introduces the Odin service, its design and implementation. Section 4 describes its evaluation using a small serious game as case study. Finally, Section 5 summarizes the contributions of this research.

2 Game Services

The video game industry is one of the fastest growing sectors in the worldwide economy [8]. According to the research company Gartner, global video game sales will reach $111.1 billion in 2015, due in part to the growth in mobile game play and the recent release of the new generation of game consoles. In order to increase engagement and player retention, video games include several common features such as leaderboards and achievements. The massive use of this approach and the impressive growth of players led to the concept of gamification as a service, later materialized in Game Backend as a Service (GBaaS). The approach is simple. Instead of replicating the implementation of the game features in each version of the game for various platforms, GBaaS adhere to a service oriented architecture providing cross-platform game services that lets you easily integrate popular gaming features such as achievements, leaderboards, remote storage and real-time multiplayer in mobile games.

While the concept of ”winners and losers” can hinder the motivation of students [7], gamification is currently being applied with relative success in e-learning [1, 6]. The integration of game concepts in learning environments helps students to remain focused and to fulfil their course goals. However, the implementation of gamification in these domains is often trapped in ad-hoc solutions or supported by specific platforms (for instance, the badges in Moodle), instead of using approaches such as those provided by GBaaS.

In the following subsections we briefly summarize the main common game features that can be applied to the teaching-learning process. Then, we compare six GBaaS regarding social and technical features. This study is part of an effort to select an GBaaS on which to base the development of a service for gamification of learning activities.

2.1 Game concepts

Games are more interesting when players are able to achieve goals and compete against other players. These features foster retention and competitiveness, and are applicable also in the gamification of e-learning activities. The following list enumerates the most common game concepts:

Leaderboards are databases that keep scores. They allow users to post their scores in a game and compare themselves with other players' scores. They measure the success of a player in a game.

Achievements are goals/challenges set in a game that players managed to accomplish. Achievements give players a motivation to keep playing, to earn as many as possible, and a way compare themselves with other players. The fulfilment of a goal may enhance the status of the player or unlock access to other levels, for instance.

Multiplayer is a play mode that allows several players to simultaneously cooperate or compete in a game. This feature supports a range of other subfeatures, such as challenges, where players compete each other on either a score challenge or an achievement challenge, and matchmaking games for real-time, turn-based, or self-hosted matches.

Saved games allow the remote storage (in the cloud) of game data, for instance, the state and the players progress in the game.

Quests are periodic game challenges that players can complete to earn rewards. This way, developers can launch periodic challenges to their gaming communities.

Gifts allow players to send/request game resources or items to/from friends (for instance, in their Google+ circles).

Matchmaking automatically sets up game matches and finds opponents based on parameters set by the game developer. Usually only a specific number of players can be matched at the same time.

2.2 Game Backend Services

A **Backend-as-a-service (BaaS)** is a cloud computing service model acting as a middleware component that allows developers to connect their Web and mobile applications to cloud services via application programming interfaces (API) and software developers' kits (SDK). BaaS features include cloud storage, push notifications, server code, user and file management, social networking integration, location services, and user management as well as many other backend services. These services have their own API, allowing them to be integrated into applications in fairly simple way [3].

A **Game-Backend-as-a-Service (GBaaS)** is a subset of a BaaS that includes cross-platform solutions for the typical game concepts identified in the previous subsection. During the development process of a game (or a generic application) developers must choose between building their own back-end services or using an available game back-end platform. This last option is usually preferred since GBaaS include several services specifically tailored for game development. These services allow developers to focus on the game logic by freeing them from implementing boiler plate features.

The following subsections compare several GBaaS according to their social and technical features. Given the number of GBaaS found (32) it would be impracticable to study them all. Therefore, eight GBaaS were chosen: Google Play Game Services, Yahoo Bakend Game Service, GameUp, Flox, GameSparks, Fresvii, Kumakore and Photon. These features are summarized in Table 1.

Social game features The studied GBaaS provide developers with social game services accessed through cross-platform API. These features make the gameplay more competitive and collaborative, and improve social engagement.

Analysing Table 1 one concludes that almost all GBaaS supports leaderboards, multiplayer game mode and cloud storage. Other features such quests and matchmaking are not yet widely supported, probably due to their novelty.

Table 1. Social and Technical game features

Types	Features	Google	Yahoo	GameUp	GSparks	Fresvii	Photon
Social	Leaderboards	yes	no	yes	yes	yes	yes
	Achievements	yes	yes	yes	yes	no	no
	Multiplayer	yes	yes	no	yes	yes	yes
	Save Data	yes	yes	yes	yes	yes	yes
	Quests	yes	no	no	yes	no	yes
	Gifts	yes	yes	no	yes	no	yes
	Matchmaking	no	no	yes	no	yes	yes
Technical	Auth	G+	Yahoo Face	Face	Face Twitter	Face	Face
	WS	REST	-	REST	REST	-	REST
	Res. format	JSON	-	JSON	JSON	-	JSON
	Platforms	Android iOS C++	ActionScript iOS Android C# Unity	Android iOS Unity	ActionScript C++ Cocos2D JavaScript Marmalade Unity	Android iOS Unity	Android .NET Unity

Technical game features The studied GBaaS offer cloud services through API and SDK to various platforms. Regarding authentication almost all GBaaS use the same strategy. Before the game can make any calls to the game services, it must first establish an asynchronous connection with the backend servers and authenticate within the game services. Some GBaaS requires that the players have an account on specific backends (GPGS requires that users have a Google account). Others, such as GameSparks, provides a simple mechanism that allows games to implement social login without any additional code, allowing gamers, for instance, to sign in using a Facebook or Twitter account, and start playing.

The majority of the GBaaS provides a HTTP RESTful API. The format of the data in all HTTP store operations (PUT and POST) are required to be valid JSON. All response data from the GBaaS comes back also in JSON format. Regarding the REST API reference, the authors opinion is that GPGS is the most complete and better documented API.

In complement to the REST API most GBaaS support also mobiles. There are examples of SDKs for Android, iOS, and even FirefoxOS (GameUp) mobile native apps. Game engines are also supported and most GBaaS offer SDKs for major game engines such as Unity, and also for cross-platform game development tools such as Marmalade and Cocos2D.

3 Odin

This section describes Odin, a gamification RESTful Web Service to be used by educational institutions. It provides (1) score submissions, (2) leaderboards listing, (3) quests for players, (4) awards to players for in-game accomplishments as well as some minor services to manage institutions, players, leaderboards, quests and achievements.

Odin is based on a standard gamification API but has a different approach regarding authentication. Institutions, rather than end-users, are the ones that require authentication. Once an institution is authenticated, Odin grants it permission to manage scores, quests and achievements in its users.

The next subsections present the architecture of Odin and its main components, and describe its data model and service API.

3.1 Architecture

Odin is a RESTful Web Service that allows institutions to consume gamification resources from their web applications. The web applications initialize sessions in Odin through authentication built on top of OAuth2 authorization protocol. Then they requests particular actions to the server identified by a specific URI and an HTTP method such as POST, GET, PUT or DELETE.

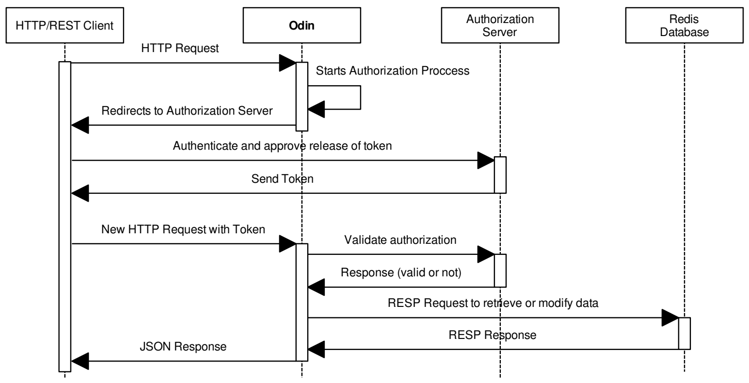
HTTP/REST Client
Odin
Authorization Server
Redis Database
HTTP Request
Starts Authorization Proccess
Redirects to Authorization Server
Authenticate and approve release of token
Send Token
New HTTP Request with Token
Validate authorization
Response (valid or not)
RESP Request to retrieve or modify data
RESP Response
JSON Response

Fig. 1. Sequence diagram representing a common request to Odin

Figure 1 presents a sequence diagram that summarizes the interactions of Odin with other systems when a request is made by the client. Firstly, the HTTP request made by the client is subject to a security filter that checks if the institution is authenticated. If the institution is not authenticated or authorized to access Odin resources it is redirected to the authorization server where it will authenticate and approve the release of a token with the authorization proof. The generated token (with expiration time) is sent to the client and it (client) presents the access token to Odin.

When the client is authenticated and authorized, it is passed to the JAX-RS REST interface implemented using Jersey (described in the next subsection) and forwarded to the mapped resource. From the resource layer it is forwarded to the service layer, passing through a security layer which intercepts it to check authorization and roles, ensuring that only authorized institutions have access to the services.

The service layer responds to the request with the data persisted on Redis (described in the next subsection) through the Jedis client (using REdis Serialization Protocol) and Ohm library implementation for Java. The response sent to the client is a JSON object representing the resource type modified or requested by it (each resource type may have one or more data representations). Whenever a fresh token is needed, the client can request it from the Authorization Server.

3.2 Frameworks and Tools

Odin uses Jersey, an open-source framework that is the reference implementation of the Java API for RESTful Web Services, extending it with additional features and utilities to further simplify RESTful service. Among other features, Jersey provides a *Core Server* to build RESTful services based on annotations, support for JSON and to the Java Architecture for XML Binding, as well as a *Core Client* to easily create a client that can communicate with REST services.

Data storage relies upon Redis NoSQL database that provides an open-source and advanced key-value storage and cache solution. It is an high performance alternative to the traditional Relational Database Management Systems (RDBMS) [5] to store and access large amount of data. Redis is sometimes described as a data structure server since keys can contain strings, hashes, lists, sets and sorted sets. As a NoSQL database it focus on performance and scalability rather than in guaranteeing the atomicity, consistency, isolation and durability (ACID) properties. Redis was selected for backend due to its hability to store large amounts of non critical data very efficiently.

In order to integrate Redis in Odin the data layer resorts to the Jedis client, as well as of an object-hash mapping library, named JOhm, to store and retrieve objects from Redis with a higher level of abstraction and thus simplicity. JOhm is the Java implementation of the well-known Ohm library and aims to be minimally-invasive, relying only on reflection aided by annotation hooks for persistence.

3.3 Data Model

The data model of Odin consists of seven main entities: institution, player, leaderboard, score, quest, achievement and session, related as denoted in the UML class diagram of figure 2.

An institution is the entity that manages games and all related data, and so it is the one which needs authentication and/or authorization. Thus, it needs to store an id and password to authenticate, and also a token to check the validity of the session. Whenever an institution authenticates a session is created and linked to it (through the *institutionId*). This session contains the creation time, last access time and a state indicator (active or inactive).

The institution needs to represent its students. As this is a gamification model they are abstracted to players, and so they will have a *playerId* that identifies him to the institution, a *displayName* that is the name to show on the

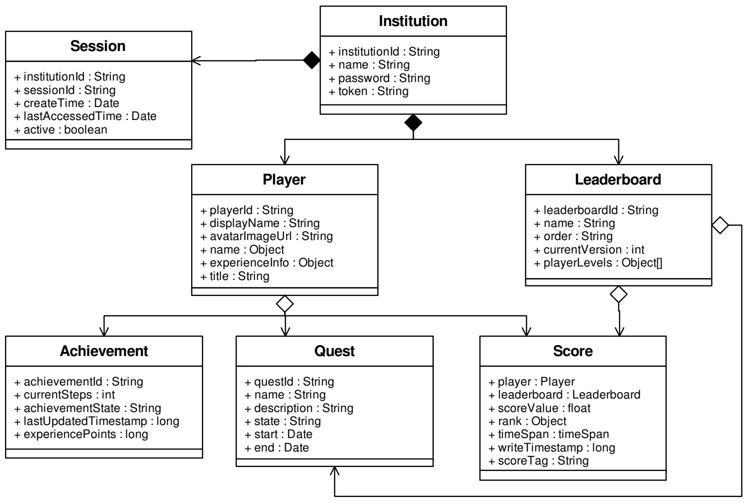

Fig. 2. Class diagram of the data model of Odin

leaderboard, a full name and a representation of his experience info with level, points acquired and needed points to level up.

As the player progresses in the game, (s)he will possibly win achievements. An achievement has a number of required steps and a state (hidden, revealed or unlocked). When a player reveals one, he receives the number of experience points associated.

A player can also accept and fulfill quests. A quest is characterized by a name, a description, a state (upcoming, open, accepted, completed, failed, expired or deleted) and a start and end date.

One of the most important parts of this model is the leaderboard. It contains more than a list of sorted scores, it contains data related to a game, such as a list of info on the levels available in the game/leaderboard. These parts are joined since it is required a single leaderboard to each game, and they depend on the existence of each other.

Scores related to a leaderboard and a player, are also stored. Each score has a floating point value, a timespan (daily, weekly or all time score), a timestamp and a rank (its position on the leaderboard).

3.4 Service API

The integration of Odin with other systems relies on REST calls to set and retrieve data. It follows the Google Web API Reference for achievements, leaderboards, players, quests and scores resources. The only differences are that all these resources URI paths are relative to gamify/institutions/**institutionId**. Also when an authenticated player is referenced in a function, it is replaced by a sub-path of the form /players/**playerId** right after **institutionId** in the resource path URI.

The institution resource is added to the set of resources. It contains the functions shown in table 2.

Table 2. Intitutions resource API reference. URIs are relative to /gamify

Function	HTTP request
insert	POST /institutions
get	GET /institutions/institutionId

The `insert` function inserts the institution given in the request body. The `get` function retrieves the institution resource given its id.

4 Evaluation

For validation of the gamification service described in the previous section, a simple multiplication game was created. This game – MathGamify – can be used by primary school children to learn multiplication tables. MathGamify generates two random numbers. The first number between 1 and the current game level and the second number between 1 and 10. Then the student/player has the opportunity to answer the multiplication value of the two numbers. The score is accumulated in the ratio of the player's level until player misses, in which case the score is reset to zero.

MathGamify acts as a tool provider to a Learning Management System (LMS). The integration of MathGamify with the LMS relies on the Learning Tools Interoperability (LTI) specification. When the LMS launches MathGamify the LTI parameters are sent as part of the HTTP POST request. On request reception MathGamify uses the LTI Wrapper [4] package to process LTI communication and extract user id, name and level. The last is a custom parameter defined on the external tool configuration of the LMS.

MathGamify consumes two types of resources from Odin: score submission and listing of scores. Once the player answers a question, MathGamify communicates the score to Odin, using Jersey Client to issue the REST call, and the grade to the LMS using LTI. This grade is a value between 0 and 1, calculated by the following way: if there is a custom parameter `custom_max_score` then it is the score divided by `custom_max_score`, otherwise it is the number of correct answers divided by the total number of tries. When MathGamify initializes its GUI, and every time a score is submitted, the score listing is updated with the data returned from Odin.

One of the key components is the LTI Wrapper that implements both sides of the LTI communication. This component receives LTI requests from LMS and issues LTI requests to LMS.

The GUI component of MathGamify was developed using Google Web Toolkit (GWT), an open source Java software development framework that allows a fast development of AJAX applications in Java. The GWT code is organised in two main packages, the server and the client. The server package includes all the service implementations triggered by the user interface. These implementations are responsible of (1) the logic of the game, (2) communication with Odin and (3) communication with LMS through LTI wrapper. The selected LMS was Moodle 2.8 .

The implementation of MathGamify demonstrates the efficacy of the proposed approach in coping with the extra requirements of a serious game integrated in a typical e-learning ecosystem, where authentication is provided by an LMS. To complement its validation, Odin was also tested regarding its efficiency.

The latency of the Odin service was tested in two of its functions: (1) submit a single score and (2) list all scores in a leaderboard. Each test consisted of 1000 samples of calls to the same function, and all numbers stated below are averages per sample.

Initially the tests were run locally on the same machine as the Odin server, using Grizzly Test Container provided by Jersey, so it had no network latency. The average time to (2) was around 40 ms (leaderboard had 6 scores when the test was running). In the worst case it took 461 ms. The test (1) spent an average time of 22 ms and the worst case took 385 ms.

The same tests were repeated on an external server. During these tests an average network latency of 23 ms was observed. In this setting test (1) consumed an average time of 67 ms. The average time to (2) was 587 ms (the leaderboard had 1000 scores).

The tool used to measure time spent was ContiPerf, a lightweight testing utility that allows the user to easily turn JUnit 4 test cases to performance tests. It is base on annotations as the JUnit 4's test configuration.

5 Conclusions

Game concepts and mechanics are an useful way to engage students in e-learning activities. These kind of features are already provided by game backend services that can leverage on their authentication services and massive user base. However, gamification services that rely on external authentication are not adequate for e-learning systems that already operate on a single sign-on ecosystem.

Odin is a gamification service developed for requirements of e-learning systems. It was designed to authenticate clients rather than end-users and thus can be integrated with the e-learning systems typically found in educational institutions.

The MathGamify system is a proof of concept, that illustrates how serious games acting as tool providers for an LMS interact with the services of Odin. The authors plan to integrate Odin in a learning environment for solving programming exercises.

Odin itself will be subject to improvements. The current version provides web services for exposing the gamification service to clients. The next version will provide also a web interface to register institutions and allow them to manage their resources.

Acknowledgments. Project "NORTE-07-0124-FEDER-000059" is financed by the North Portugal Regional Operational Programme (ON.2 O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT).

References

1. Burguillo, J.C.: Using game theory and competition-based learning to stimulate student motivation and performance. Comput. Educ. 55(2), 566–575 (Sep 2010), http://dx.doi.org/10.1016/j.compedu.2010.02.018
2. Hamari, J., Koivisto, J., Sarsa, H.: Does gamification work?–a literature review of empirical studies on gamification. In: System Sciences (HICSS), 2014 47th Hawaii International Conference on. pp. 3025–3034. IEEE (2014)
3. Janssen, C.: Backend-as-a-service (baas)". Tech. rep., Techopedia, http://www.techopedia.com/definition/29428/backend-as-a-service-baas (2014)
4. Queirós, R., Leal, J.P., Campos, J.: Sequencing educational resources with seqins. Computer Science and Information Systems 11(4), 1479–1497 (2014)
5. Seeger, M., Ultra-Large-Sites, S.: Key-value stores: a practical overview. Computer Science and Media, Stuttgart (2009)
6. Siddiqui, A., Khan, M., Akhtar, S.: Supply chain simulator: A scenario-based educational tool to enhance student learning. Comput. Educ. 51(1), 252–261 (Aug 2008), http://dx.doi.org/10.1016/j.compedu.2007.05.008
7. Vansteenkiste, M., Deci, E.L.: Competitively contingent rewards and intrinsic motivation: Can losers remain motivated? Motivation and Emotion 27, 273–299 (2003), http://dx.doi.org/10.1023/A:1026259005264, 10.1023/A:1026259005264
8. Zackariasson, P., Wilson, T.: The Video Game Industry: Formation, Present State, and Future. Taylor & Francis (2012), http://books.google.pt/books?id=lgiQNdc-DOwC