Journal of Ambient Intelligence and Humanized Computing (2020) 11:961–977
https://doi.org/10.1007/s12652-019-01203-7

ORIGINAL RESEARCH

A real-time service system in the cloud

Aneta Poniszewska-Maranda[1] · Radosław Matusiak[1] · Natalia Kryvinska[2] · Ansar-Ul-Haque Yasar[3]

Received: 15 August 2018 / Accepted: 28 November 2018 / Published online: 5 February 2019

Abstract
Recently, we have witnessed unprecedented use of cloud computing and its services. It is influencing the way software is built, as well as company' resources such as servers, workstations or generally hardware are used. This paper aims to examine the benefits of cloud usage to support real-time service systems, using the Salesforce platform. First, we explore the meaning and the role of cloud computing for the real-time service systems efficient functioning. Then, we build a service management platform for the Polish Billiards and Snooker Association (PBSA), based on a real-time system located in the cloud. This way, PBSA managers are able to complete their tasks in this system on-demand. Moreover, it is set up as a private cloud to grant access only to the snooker organization employees.

Keywords Cloud computing · Real-time service system · Salesforce platform · SaaS cloud

1 Introduction

Ubiquitous development of Information and Communication Technologies (ICTs) involves the evolution of commonly adopted computer systems and applications. One of the fastest growing trends in the IT environment is the *cloud*. It is influencing the way of the software building, as well as companies or end users exploit resources such as servers, workstations or generally hardware, which in turn can become deprecated quickly. *Cloud* itself can be referred to as *cloud computing* or public cloud; e.g., online disks designed for data storage and easy access to it. Accordingly, business/private users can manage their businesses or processes using any device that has an Internet connection (Mell and Grance 2009, 2011; Jin et al. 2010; Qi et al. 2018).

From the research point of view, *cloud computing* is an interesting field as it is one of the fastest developing domains in modern businesses and classical computer science. It brings a lot of opportunities in response to on-demand access to data resources; but, at the same time, there are many concerns about the safety of such solutions. Often information that users decide to store online is sensitive. In a certain sense, there is some risk involved when using *cloud computing* as beneficiaries must trust and rely on the service providers. On the other hand, the growing number of users who takes advantage of *cloud* solutions force *cloud* providers to assure an appropriate level of security. Hence, more and more organizations decide to move their data into the *cloud* (Tang et al. 2010; Sriram and Khajeh-Hosseini 2010; De la Prieta et al. 2017). As an example, an organization that lacks a service management computer system, we analyze the Polish Billiards and Snooker Association (PBSA). It is a Polish federation representative for two sports disciplines—snooker and English billiards. Its main occupation embraces organizing tournaments, the popularization of mentioned disciplines, representation of Poland on the international arena and choosing players to the national team.

Polish Billiards and Snooker Association does not have distributed computing infrastructure. Also, the activities on behalf of this organization are not the main occupation of people from it, rather just hobby. The *cloud* approach with its advantages seems a reasonable solution to build a system that could help to manage snooker tournaments. Cloud

✉ Aneta Poniszewska-Maranda
aneta.poniszewska-maranda@p.lodz.pl

Natalia Kryvinska
Natalia.Kryvinska@fm.uniba.sk

Ansar-Ul-Haque Yasar
ansar.yasar@uhasselt.be

[1] Institute of Information Technology, Lodz University of Technology (TUL), Lodz, Poland

[2] Department of Information Systems, Faculty of Management, Comenius University in Bratislava, Bratislava, Slovakia

[3] Transportation Research Institute-IMOB Hasselt University, Hasselt, Belgium

platform enables access from personal computers or smartphones from anywhere as long as an Internet connection is available (Sadiku et al. 2014; Poniszewska-Maranda et al. 2017; Chen et al. 2018).

There are many aspects that would make the organization of such competition more efficient. For example, the main referee should have a tool that allows managing the schedule of matches and notifying players about their next match, generating match report, and generating tournament bracket automatically. Accordingly, we examine the capabilities of *cloud* in relation to the *real-time service systems*, considering *Salesforce* platform.

This paper is organized as follows. Section 2 describes the idea of *real-time system* and *cloud computing* common usage. This section also describes cloud, key characteristics, deployment and service models. Section 3 reveals the information about *Salesforce* platform—what it is for, and what tools it offers to build our proposed system. Section 4 deals with the building of the proposed system, including the development of the requirements, architecture and logic layer. Section 5 provides a summary of the results and proposed future work.

2 Cloud computing for real-time service systems

Polish Billiards and Snooker Association (PBSA) service management platform is developed as a real-time system located in the cloud. It allows PBSA managers and employees to call up this system on-demand. It is deployed as a private cloud with restricted access to the employees of the snooker organization only. This section describes a real-time system functionality and its usefulness for our proposed management system. Then, a categorization of cloud capabilities in relation to the research problem is discussed.

2.1 Real-time systems

The term *real-time system* sometimes can be easily misunderstood. One might probably think of a complex system that processes data, operates and responds to the user almost immediately. In fact, this concept is much wider. The crucial thing is to distinguish the difference between *real-time system* and computer real-time system (Shin and Ramanathan 1994). A definition proposed in (Laplante 1994; Schoch and Laplante 1995; Laplante and Ovaska 2011) states that not only computations correctness is important in *real-time systems*, but also the fact that they follow events happening in real time. In this case, it means that the essential requirements are punctuality (meeting time constraints) and continuous operation. According to a concept proposed in (Kopetz 2011), such systems are divided into the set of subsystems, referred to as *clusters* (Kopetz 2011). Figure 1 depicts the components of *real-time systems*.

Based on this view, a computer real-time system is just one of the *clusters* of a bigger system where computations take place. Hence, the relation through the man–machine interface is obvious—a man operates a computer that processes data related to a controlled object. An interface, in turn, consists of sensors and actuators that process physical signals and transfer them to or receive them from the computational cluster (Gomaa 1993; Liu et al. 2000; Kopetz 2011).

Real-time systems can be divided into two categories, namely: *hard* and *soft*. The most crucial factor here is time. A computer real-time system should operate within a requested time constraint referred to it as a *deadline*. *Hard* type of *real-time systems* means—every *deadline* must be met; otherwise, the system can result in a total failure. Such breakdown could cause a disaster, even for humans. Such disasters can happen in chemical plants or in air traffic control, which are in turn *real-time systems* (Mattai 1995; Kopetz 2011; Laplante and Ovaska 2011). On the other hand, *soft* type of *real-time systems* does not cause any danger when the *deadline* is not met. When a failure occurs in this case, a system might become just annoying for the user or may result in some financial losses. A good example to illustrate this problem is an online reservation system. Missing a time constraint would result in unsuccessful operation, but no danger for people (Krishna 2001; Kopetz 2011; Laplante and Ovaska 2011).

Another classification of *real-time systems* involves *event-triggered* and *time-triggered* systems. A *trigger* can be defined as an event that entails further actions or computations in the computer. Hence, in *time-triggered* systems, any activities taken by it happen at a predefined earlier moment. This makes the whole system inflexible since it handles only events that had been defined before. On the other hand, actions in *event-triggered* systems are started immediately after any event other than set earlier to happen at a given time triggers the system (Kopetz 2011; Klein et al. 2012).

Within the context of this work, a system is developed for practical purposes that can be defined as a soft real-time system. In case of total failure, naturally there is no a significant danger for people or surroundings. It is a tool designed for improving the efficiency of management sports tournaments. Therefore, if it doesn't work properly, people in charge have to operate just like they are doing it now—without any automation. Hence, the time constraint involved here should be such that the work with this system is fluent, approximately 2–3 s per action invoked by a user. It is also an event-triggered system, because it reacts to the results on the tournament. As this is not so complicated, from the theoretical point of view of real-time systems, the instrumentation interface is rather neglected. This is because we can not define the controlled object as one physical object.

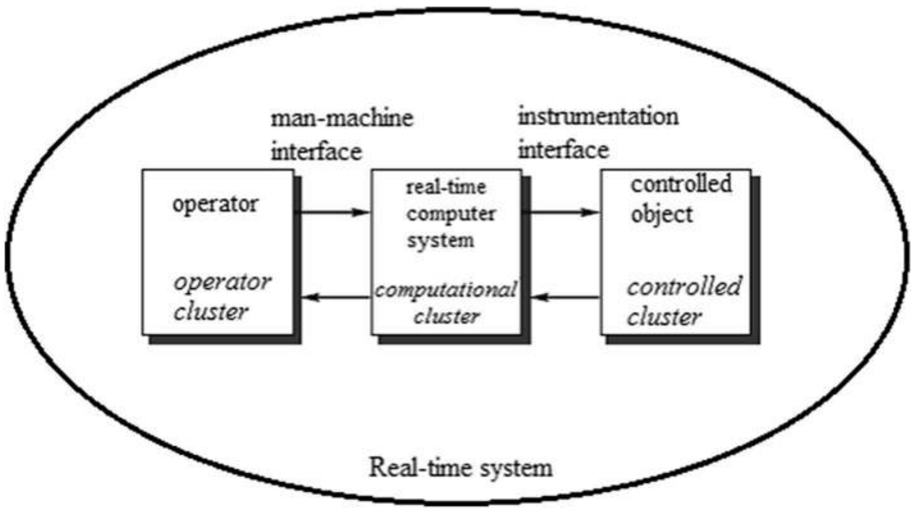


Fig. 1 Decomposition of the real-time system (Kopetz 2011)

In this case, it rather refers to the tournaments as a whole—matches, calculation of results and registration process.

2.2 Cloud computing

Mell and Grance from National Institute of Standards and Technology (NIST) defined cloud computing as a model that provides end users with access from any device, which has an Internet connection, to a shared set of resources such as servers, applications or services (Computer Security Division 2018; Mell and Grance 2009, 2011). Therefore, the most crucial fact in the cloud computing concept is that the software being used is not installed on a device from which it is accessed. Appropriate services do not involve companies' infrastructure, they are available over an Internet connection, ready to use (Leavitt 2009; Tang et al. 2010). One of the first companies, who started to offer cloud services over the Internet was *Salesforce*. A further example was *Amazon Web Services* that enabled users the possibility to store data online and perform computations (Barry 2003; CRM Software & Cloud Computing Solutions 2018).

NIST provides four deployment models: private cloud, community cloud, public cloud, and hybrid cloud (Angeles 2013; Computer Security Division 2018; Barry 2003; Sriram and Khajeh-Hosseini 2010; Mell and Grance 2011). Besides, it considers three main service models. These are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Service models and deployment models are strictly connected with each other. Besides, it is important to remember that no matter whether the end product is delivered in IaaS, PaaS or in SaaS—it can be built on any of deployment models (Cusumano 2010; Sadiku et al. 2014; Sultan 2014). In the following, we provide some implementation examples of cloud service models.

In the first model, Infrastructure as a Service (IaaS), a customer is given computing resources such as virtual server space, networking, operating systems so that it can deploy and run the software. Companies such as Amazon, Rackspace or GoGrid are examples of IaaS providers. Resources given to the customer are distributed across data centers, which are maintained by the provider. Customers, in turn, can work in virtualized components and deploy their own platform. The reason for employing the IaaS model in companies is that it allows moving expenses of managing hardware to the cloud provider. Infrastructure as a Service allows the consumer to build scalable system as resources based on their needs (Angeles 2013; Computer Security Division 2018; Barry 2003; Sriram and Khajeh-Hosseini 2010; Mell and Grance 2011).

The next service model is Platform as a Service (PaaS). The customer is again given the cloud infrastructure and can deploy their own system into the cloud utilizing tools available by the cloud provider such as programming languages, libraries or database management systems. Moreover, available tools do not exclude the usage of libraries and tools from other integrated sources. Examples of PaaS providers are Google App Engine, Microsoft Azure, Force.com and Amazon. Cloud infrastructure is maintained by the cloud provider, where application programmers can fully concentrate on the system development. Moreover, the tools provided in the PaaS model often enable developing applications without the knowledge of programming. For example, Force.com platform allows building parts of a system using one-click functionalities (Angeles 2013; Computer Security Division 2018; Barry 2003; Sriram and Khajeh-Hosseini 2010; Mell and Grance 2011).

The last model mentioned in NIST definition is Software as a Service (SaaS). In this case, a customer does not build its own system but makes use of a ready system prepared

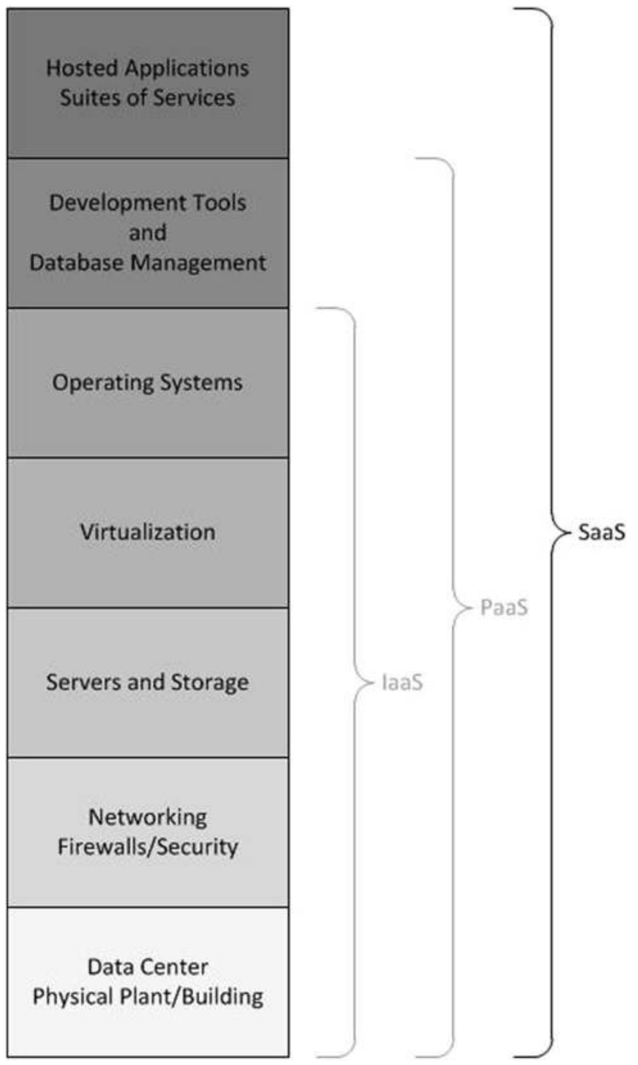


Fig. 2 Elements of cloud service models (Computer Security Division 2018; Mell and Grance 2011)

by the cloud provider and running it in the cloud. Examples of SaaS providers are Salesforce, Microsoft Online Services, and Google Apps. Applications created by SaaS providers are accessible from any device with Internet connection. For example, a user can download a program from Google and use it on his/her smartphone, or a company with Salesforce licenses can take the advantage of ready to use applications from Salesforce's marketplace—AppExchange (Computer Security Division 2018; Sriram and Khajeh-Hosseini 2010; Mell and Grance 2011). Figure 2 illustrates the main elements that constitutes these models (Salesforce Developers|API Documentation, Developer Forums & More 2017).

In this paper, the real-time service management system advanced is built on two service models. The final solution delivered to the Polish Billiards and Snooker Association, from the point of view of this organization, is given in Software as a Service model since it is ready to use the system in the cloud. During its development, Force.com platform provided by Salesforce is used, which involves another model; i.e., Platform as a Service. Issues related to the Salesforce functionality and its PaaS platform are described in the next Section.

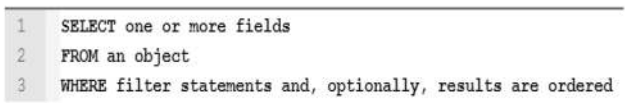
3 The functionality of Salesforce cloud service delivery platform

This section provides a detailed description of Salesforce components that are used to build our proposed system. Salesforce mainly provides customer relationship management (CRM) software, primarily cloud-based. Salesforce has a high reputation among its beneficiaries as it is confirmed by numerous awards for the best CRM system given by magazines Enterprise CRM, Mid-Market CRM or being a leader in ranking of recommended CRM's led by Software Advice based on customers' reviews (CRM Salesforce 2017; Youseff et al. 2008; Benioff and Adler 2009; Trailhead 2017).

3.1 Database in Salesforce

The database represents the persistence layer of each application. In the case of *Salesforce*, users and developers are not responsible for managing this part of the system as there is no software that has to be installed for that purpose. Instead, the platform delivers the database that can be used when building the applications using *Force.com*. It is a relational type, but there are a few differences in comparison to the database management systems in traditional applications. In the latter, data is kept in *tables* which consist of *columns* defined by a particular data type. Information might be retrieved among *rows* of the concrete table. Furthermore, these tables can be associated with each other with the use of a primary key in one table and foreign key in the related table.

Object replaces *the table* in *Force.com* database. In *Salesforce* nomenclature, it is crucial to differentiated from so-called *sObject*, which can be described as an object materialized in the platform's object-oriented programming language, *Apex*. This term is introduced to differ such objects from instances of classes. Hence, the main function of *Salesforce's object* is to store the data. *Objects* can be related to each other via *relationship fields*. There are two types of them in *Salesforce*, namely: *Lookup* and *Master–Detail*. In both cases, the *ID* of the related object is saved in *the relationship* field. The first type reflects the relationship of type "one-to-one" or "one-to-many" and creates a link between two objects. Among standard *Salesforce* objects, there is always a relationship of *Lookup* type between *Account* and



```
SELECT one or more fields
FROM an object
WHERE filter statements and, optionally, results are ordered
```

Fig. 3 Structure of SOQL query (Salesforce developers)

Contact. The latter has the *Lookup* field that stores the *ID* of a related object, in this case—*Account*.

The *Master–Detail* relationship is used when two objects are tightly bounded with each other. Deletion of parent record in this relationship entails deleting all child records. This kind of relationship might reflect "many-to-many" relationship. It happens with the use of so-called *junction object* that has set up *Master–Detail* field to two other objects (Salesforce developers). *Force.com* platform offers a special language dedicated to retrieving data from its database, called Salesforce Object Query Language (SOQL). Its statements are very similar to the well-known language of relational databases. The basic structure of SOQL query is presented in Fig. 3.

3.2 Apex

The *Salesforce* platform is using *Apex* which is an object-oriented programming (OOP) language. Its syntax strongly bears resemblance to Java as it is based on its conceptual principles. Additionally, it supports the OOP paradigm such as classes, inheritance, and interfaces. *Apex* is executed and saved on the *Force.com* platform. The working principles of this language is presented in Fig. 4.

Apex enables the conditions and statements available in the majority of programming languages, such as variables, constants, if-else statements, loops, and array notation. However, in the case of this language, an array is the same as one of the collections, i.e. list. Other collections available in Apex are sets and maps. What makes Salesforce's programming language unique is that the Cloud is the place where the code is executed and compiled. Apex also enables SOQL calls and assigning its results to the chosen collection. Moreover, it supports Data Manipulation Language (DML) operations, which can insert, update or delete the records from the database.

3.3 Visualforce: client-side language

Visualforce is a framework that enables web development within the Salesforce platform as such creating customized user interfaces. All interfaces are built using of this language may consist of Visualforce tags, starting from prefix 'apex' (e.g., the starting tag <apex:page>), plain HTML and optionally styling elements such as css elements that can be uploaded into Salesforce. Moreover, Visualforce can comprise JavaScript. The working principle of this framework is depicted in Fig. 5.

Each interface has its own unique URL address through which it can be accessed. Then, it can interact with the system logic, defined in *Apex* classes (custom controllers) or triggers or in standard controllers, enabling automatic access to data in *Salesforce*. Custom *Visualforce* interfaces might be used in various places inside the platform. To give an example, standard *Salesforce* interfaces for editing or creating a new record can be overwritten by *Visualforce* ones. They can be also accessed through the custom buttons created by a user. The other options are embedding custom interface into a standard layout or to open it through the custom tab (trailhead and Salesforce developers).

4 Real-time service system based on the Salesforce platform

This section provides a practical example of real-time system employment into cloud computing. It is a system we developed with the use of *the Salesforce* platform named *Top 16 Manager*. The main goal of this system is to examine

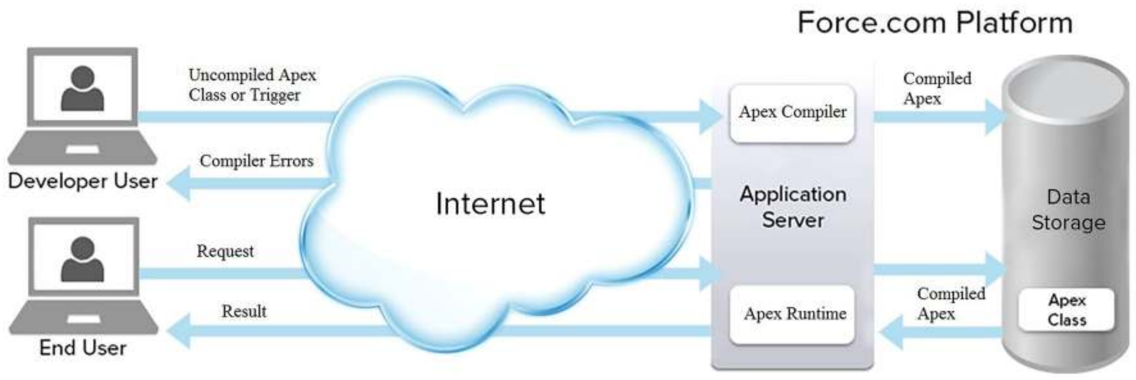


Fig. 4 Structure of SOQL query (Salesforce developers)

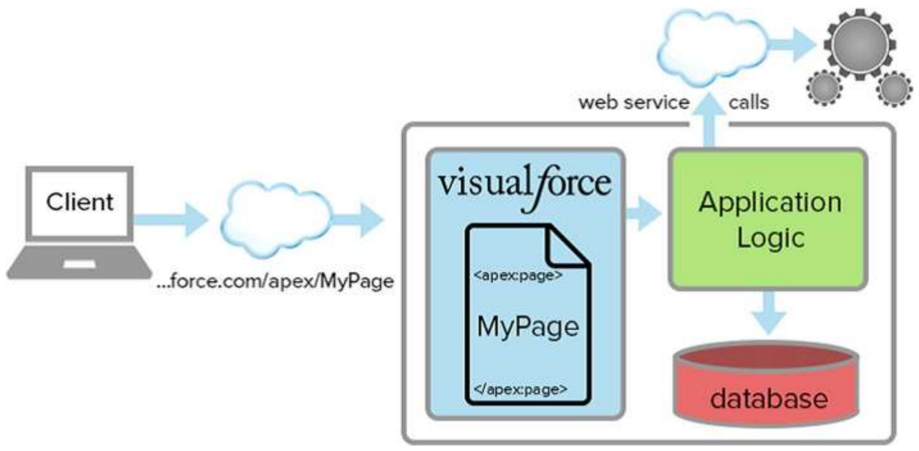


Fig. 5 Principles of working of visualforce (CRM Salesforce; trailhead; Salesforce developers)

Salesforce platform capabilities in the context of real-time systems and cloud computing. For that purpose, the system which can be classified as the soft real-time system is created in the private cloud, taking advantage of Force.com tools and programming languages. One of the best advantages of Salesforce is that customizations can be achieved by using few administration tools offered by Force.com platform. Applications in Salesforce are mainly built for business purposes, where there is a possibility to define Email Alerts with their contents defined in an Email Template. This can be invoked using other Force.com tools, namely: Workflows and Process Builder. Both of them are used to automate the processes used in Salesforce application. Their creation starts from defining which object is going to entail further actions. For example, these include updating the field value on defined records or sending an email according to Email Alerts indicated earlier. Actions set in these tools are triggered only if they meet certain criteria given by the user who creates either a workflow or a process. These actions might be either immediate or time-dependent. The difference between these two tools is that the Process Builder allows updating child records what is not allowed in workflows. All described processes are created only by clicking and setting right options, no code is involved in such cases.

Salesforce may have many users; therefore, there is a need to define who can access what. It can be achieved by setting appropriate profiles using another Force.com tool which does not involve any technical knowledge. Users' profiles are set of permissions and their settings regulate which resources a particular user can access. For example, they can control view, update, create or delete the permissions for each object defined in Salesforce, which fields and buttons are visible for the user or which Apex class or Visualforce interface can be executed by a particular user (Salesforce developers). Although Salesforce is 17 years old and many users claim that the design of the platform is old-fashioned, many functionalities are stated that makes it so powerful to build the CRMs in the cloud. However, with the release of the Salesforce1 mobile application, platform's creators decided to move its design into the desktop version of Salesforce. This is also done to meet the expectations of users for which the design of the system is its important part. For that purpose, Salesforce launched lightning experience; a new graphical user interface of the whole platform. What has changed apart from the overall design of the platform is, for example, navigation menu, record layouts, list view, and dashboards. Salesforce has also released Salesforce lightning design system (SLDS); a CSS stylesheet consistent with lightning experience principles. It is also possible to develop the interfaces in Visualforce and apply lightning styling on them. Except for improved look of the platform, lightning entails changes in the programming conventions in Salesforce. The new user interface framework is released in order to develop the applications on the desktop and mobile versions of the platform. It is called lightning components. Just as in Salesforce classic, the server side is handled by Apex. However, client-side in case of lightning components is served by JavaScript. It enables an interaction with custom controllers and with the database. Lightning components might be accessed in the system through the tab, they can be the part of custom interfaces or they can be defined under quick action which may be applied to each record. Typically, a single component consists of a few elements put in definition bundle. It comprises of a component in which a user interface is defined and JavaScript handlers: controllers and helpers. Optionally, CSS definition of a component might be included (Trailhead).

Salesforce is undoubtedly an interesting platform as it is leading CRM software provider as well as from the cloud computing research point of view. It connects its two service models (PaaS and SaaS), as the final product used by customers is ready to use the software while developing applications in Salesforce that involves Force.com platform. A big advantage of Salesforce is the fact that except for technical tools, programming languages dedicated specially for this platform, it also has ones that require setting correct options and processes within a few clicks. It is worth to notice current development of the platform, especially with the release of lightning as every upgrade brings new functionalities and possibilities of application creation.

4.1 System requirements

The main function of the *Top 16 Manager* system is the management of the snooker tournaments *TOP 16* organized periodically by *Polish Billiards and Snooker Association*. Such an event is a challenge from the organizational point of view. Usually, there is only one person, i.e., the main referee who is responsible for the efficient conduct of the tournament. It embraces managing the start time of each match, their order, assigning referees and placement of matches on available tables. What is more, the main referee must inform every player about their incoming match, which sometimes quite challenging. The functional requirements for the Top 16 Manager are as follows: putting match scores to the system, calculation of table in the group stage, generation of bracket for the knock-out phase, possibility of referee, table and status assignment for all matches, match report generation, e-mail notification for players about the next match, and mass e-mail message to players taking part in the tournament.

The basic feature of the *Top 16 Manager* is the ability to put match scores to the system and calculate the table of each group in the tournament. The system automatically draws groups for the tournament and presents the list of matches. After the end of the group stage, it is possible to generate a bracket of the tournament based on scores from the first phase. In this way, quarterfinals, semifinals and final games are presented in the system. When a match of the knock-out phase is finished, the bracket is automatically updated. The panel of tournament matches is constructed in such a way that the main referee can easily assign to the single match the following: a referee, table and status, which makes managing them more efficiently as all required information is in one place. Figure 6 illustrates the use-case diagram with all the functional requirements of the *Top 16 Manager*. It embraces only one actor that is the main referee as it is the only user in this system due to limitations of the developer edition. With a license of multiple users, it is possible to add to this diagram other actors, such as players.

4.2 Top 16 Manager system architecture

The architecture of any *Salesforce* application is defined as a *multitenant architecture*. It means that users of the platform share the same resources and tools provided by *Force.com*. At the same time, individual users of *Salesforce* can take advantage of the customization built on the platform especially for them. Moreover, *Salesforce* utilizes *metadata-driven* architecture. All platform customizations such as interface layouts, objects definition, workflows, processes or *Apex* code are stored as metadata. In the *Salesforce* database, no tables are created such as in traditional relational databases. What is more, the code from the *Apex* classes and triggers are not compiled. Instead, stored metadata is used by system engine at runtime to generate virtual application.

The multitenant data model of discussed platform consists of multitenant metadata, multitenant data, and multitenant indexes. Multitenant metadata comprises internal tables called MT_Objects and MT_Fields. These tables store

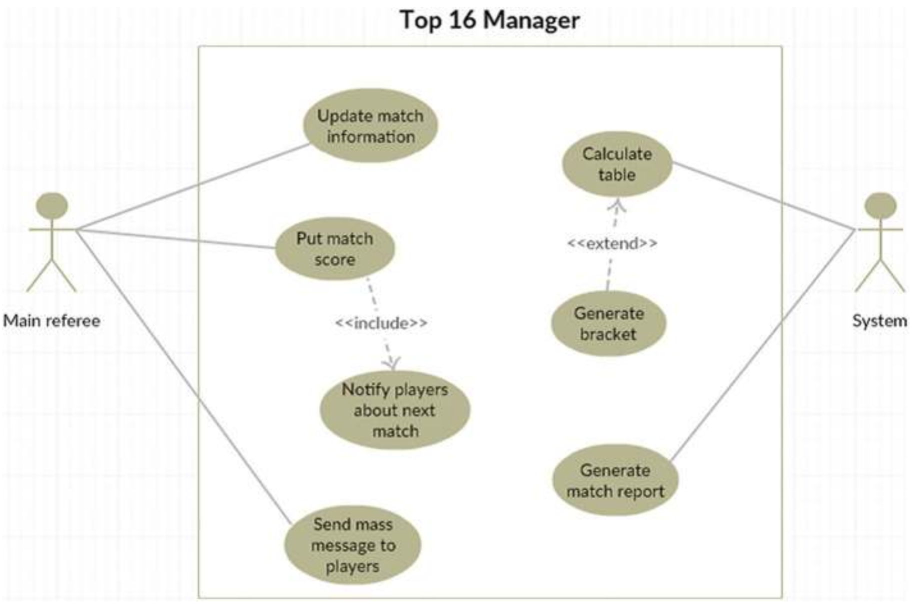


Fig. 6 Use-case diagram of Top 16 Manager

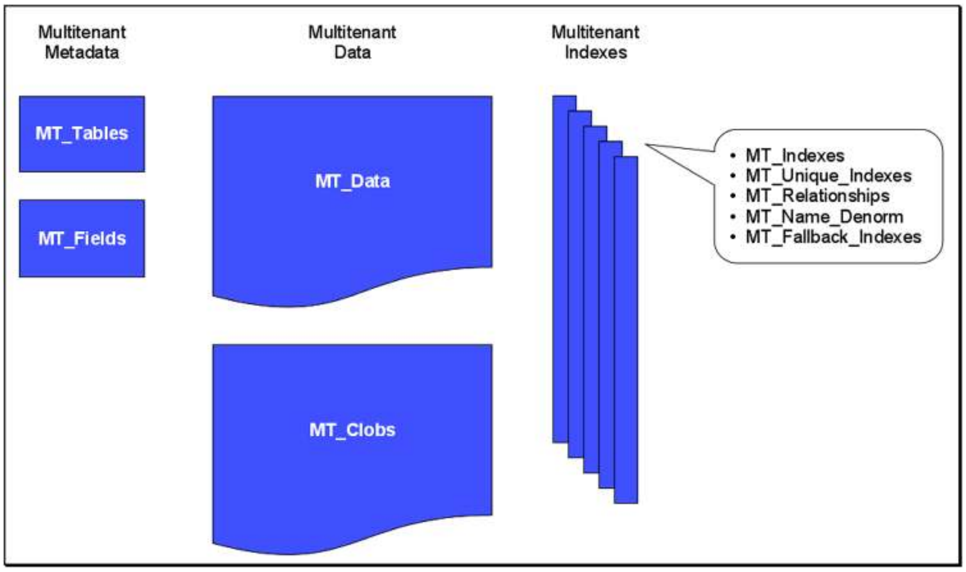


Fig. 7 Multitenant data model on Saleforce platform (Salesforce developers)

information in the form of metadata about objects together with organization ID, to which these objects belong and about fields that are defined on objects. Multitenant data, in turn, stores individuals' data inside Salesforce application that can be mapped to their objects and fields defined in MT_Objects and MT_Fields. This data type can be available on the Salesforce platform; for example, number, text, date, formula. It is also possible to store long text information as character large objects (CLOBs). Finally, multitenant indexes allow cataloging data stored in the cloud so that retrieving this data and access to it during runtime is optimized. Schema of the multitenant data model is depicted in Fig. 7.

The *Top 16 Manager* system comprises of three main components, for managing *TOP 16* tournaments, namely: Match management, Players notification, and User interface. Match management component is the most important component among others, as it is responsible for handling the basic functionalities of *Top 16 Manager*—input of match scores and calculation of tables. The logic of this component is defined in *Apex* classes—controllers for *Visualforce* interfaces, on which the main referee operates. The other component is *Players notification* that has the responsibility of sent messages to the players taking part in a particular competition. This component is not critical for the correct operation of *Top 16 Manager*. It also communicates with *Match management* component, because any notification send to players is based on the match details. The logic of players' notification component is defined in *Process Builder* as well as in *Apex* classes. The last component, *User interface*, is responsible for the arrangement of the *Visualforce* interfaces customization that overrides the standard ones, but also for all the layouts of object interfaces, related lists and search layouts. It is defined through a set-up of the *Salesforce* platform user interface capabilities. Figure 8 depicts the component diagram of *Top 16 Manager*.

4.3 Data model

To fulfill the required functionalities of the proposed system, the following five custom objects are developed: Tournament, Player, Tournament Appearance, Match, and Referee. The main developed object is the *Tournament*, which is the starting point of the system. It is related to the *Match* object via a one-to-many relationship. Every tournament is connected to it matches; at first 24 in the group stage and then 7

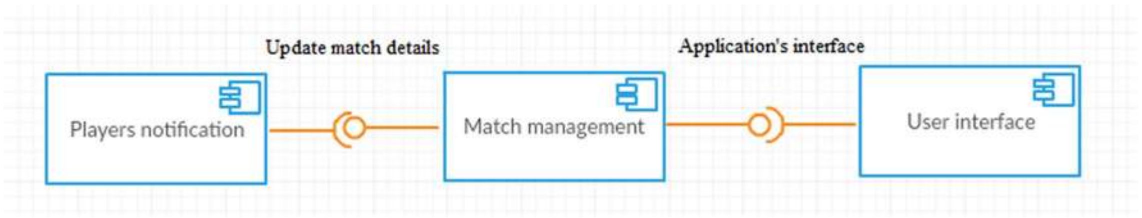
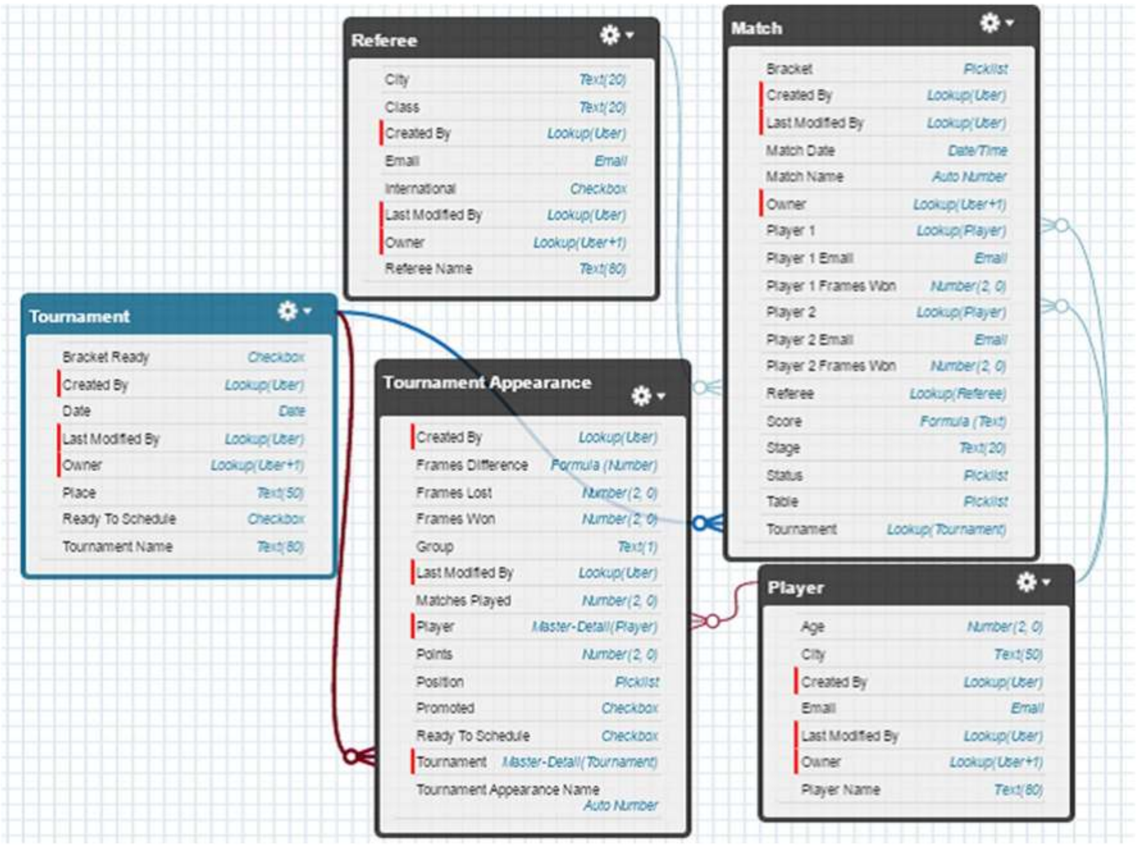


Fig. 8 Component diagram of Top 16 Manager



Referee	
City	Text(20)
Class	Text(20)
Created By	Lookup(User)
Email	Email
International	Checkbox
Last Modified By	Lookup(User)
Owner	Lookup(User+1)
Referee Name	Text(80)

Match	
Bracket	Picklist
Created By	Lookup(User)
Last Modified By	Lookup(User)
Match Date	Date/Time
Match Name	Auto Number
Owner	Lookup(User+1)
Player 1	Lookup(Player)
Player 1 Email	Email
Player 1 Frames Won	Number(2, 0)
Player 2	Lookup(Player)
Player 2 Email	Email
Player 2 Frames Won	Number(2, 0)
Referee	Lookup(Referee)
Score	Formula (Text)
Stage	Text(20)
Status	Picklist
Table	Picklist
Tournament	Lookup(Tournament)

Tournament	
Bracket Ready	Checkbox
Created By	Lookup(User)
Date	Date
Last Modified By	Lookup(User)
Owner	Lookup(User+1)
Place	Text(50)
Ready To Schedule	Checkbox
Tournament Name	Text(80)

Tournament Appearance	
Created By	Lookup(User)
Frames Difference	Formula (Number)
Frames Lost	Number(2, 0)
Frames Won	Number(2, 0)
Group	Text(1)
Last Modified By	Lookup(User)
Matches Played	Number(2, 0)
Player	Master-Detail(Player)
Points	Number(2, 0)
Position	Picklist
Promoted	Checkbox
Ready To Schedule	Checkbox
Tournament	Master-Detail(Tournament)
Tournament Appearance Name	Auto Number

Player	
Age	Number(2, 0)
City	Text(50)
Created By	Lookup(User)
Email	Email
Last Modified By	Lookup(User)
Owner	Lookup(User+1)
Player Name	Text(80)

Fig. 9 The data model of Top 16 Manager

more at the knock-out phase (quarterfinals, semifinals, and the final). This reflects the type one-to-many relation, as the single tournament has multiple matches related to each other. Match is connected with Player, which contains two Lookup fields as two players take part in a single match. It is also connected with the Referee object because a referee is always assigned to a match.

Tournament appearance object is developed to reflect the relationship of type many-to-many as multiple players can take part in many tournaments. This object acts as a junction object; it has fields of type *Master–Detail* to *Player* and *Tournament* objects. It also contains a few fields needed for calculating the table during the group stage of each tournament. Figure 9 illustrates the data model of the snooker management system, from a platform *Schema Builder* view point.

All fields marked by red vertical lines are either standard ones, such as *Created By Last Modified By* and *Owner*, or the required ones, such as *Master–Detail* fields on *Tournament Appearances* object. A standard field is also the name of each object's record. It can be either text or an auto number. *Tournament* object contains fields that describe the date and place of the single competition and technical *Checkbox* fields which are used as flags for the logic flow. Fields in the *Player* object describes the data about a player. One important information is the e-mail as it allows receiving the notifications about next matches.

4.4 Logic layer

The system development within *the Salesforce* platform involves model-view-controller (MVC) pattern. The system is divided into three separate layers for easy maintenance and to provide modularity. In this case, the *model* is defined in the standard and in the custom objects, or in *Apex* classes. *View* layer comprises of *Visualforce* interfaces and components, which are rendered on the server and displayed in the web browser. The *logic* (*controller*) layer might be customized. It is defined programmatically in *Apex* classes or triggers, but it can also use standard controllers generated by *Force.com* platform. It embraces; for example, binding of the input fields on the interface with the object, or with actions such as saving or cancel. *MVC* pattern in *Salesforce* is depicted in Fig. 10.

The logic of *Top 16 Manager* is customized programmatically in *Apex* and in various *Force.com* administration tools. The first step in the platform configuration is to define the appropriate *profile*, which could be assigned to the account

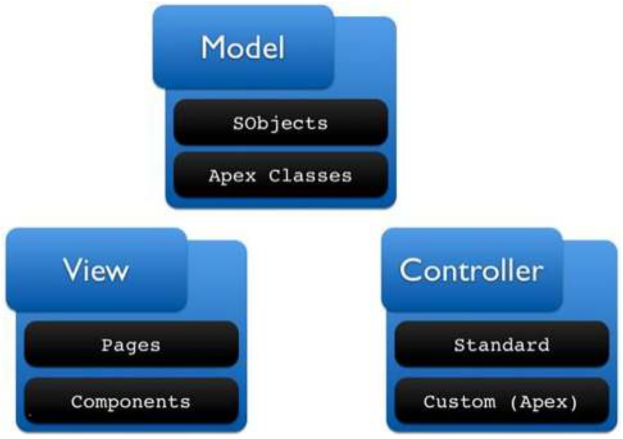


Fig. 10 MVC pattern in Salesforce (Salesforce developers)

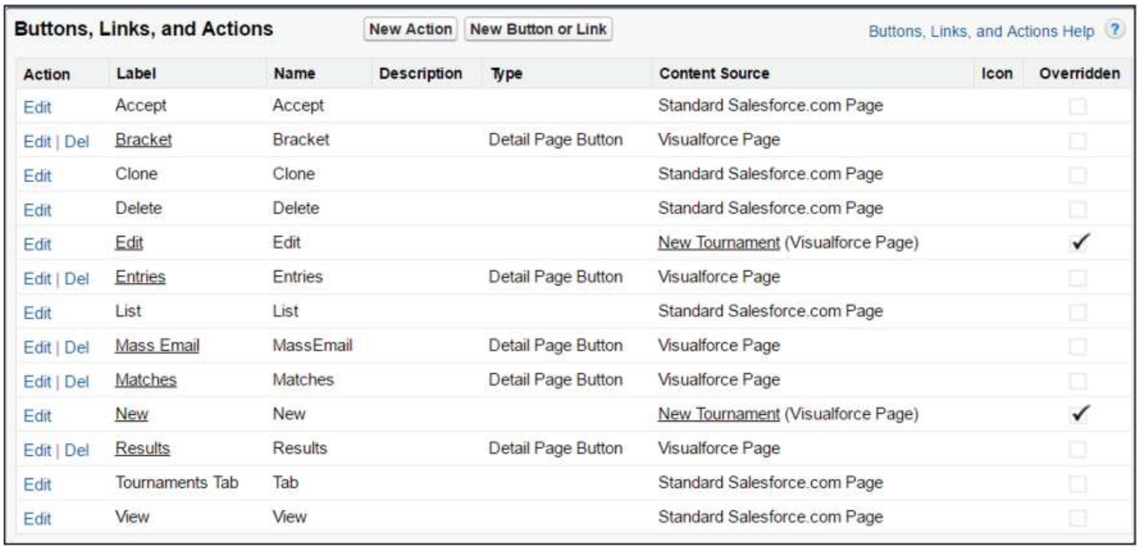
for the main referee. In the definition of this profile, all tabs for standard objects have been hidden since they are unnecessary from the system point of view. Instead, it has been set that visible tabs are those related to five custom objects used in *Top 16 Manager*. Moreover, read and write permissions have been given for the required objects. Access to every *Visualforce* interface in the system has been allowed for users with that profile.

If there is a need to add more users to the system, new profiles should be created. This concerns players as an example. In such a situation, their permissions for objects can be different. For example, they are not supposed to have edited and created access for such objects as *Matches, Referees, and Tournaments*. Due to the fact that Tournament is the main object in *Top 16 Manager*, functionalities are mainly executed from the view of records of that object. It involves creating a few custom buttons, which open Visualforce interface where a user can operate. The list of buttons belonging to Tournament object is presented in Fig. 11.

A user can enroll the first players to the tournament with the use of custom interfaces. Logic defined in *Apex* working under that interface creates the records of *Tournament Appearances* object connected with the tournament, from which this action is fired with the selected players. The system checks whether the user selected 16 players or not. If yes, then it saves it to the newly created map *Tournament Appearances*. Next, with the use of the random function it assigns to the appearance group A, B, C or D, in which a player is going to compete. After that, it saves the appearances to four lists according to their groups, which have been drawn. Finally, it inserts into the database six matches in each group, as every player must play against every other player in his/her group. Those matches assigned relation to the tournament and the status is set to *Match Ready* in the moment of insertion.

All records that are planned for insertion are first saved to the list, which is then inserted entirely, instead of inserting every record separately, based on *Apex* development guidelines. This is done to reduce the number of *Data Manipulation Language* operations, which is limited to the *Salesforce* platform. Then, actions are related to the matches of the given tournament Fig. 12 depicts an activity diagram of the process of the players enrolled in the tournament.

In the current system, *Visualforce* interface lists the games into groups. It is possible to put the score there and update its data as status, date and time, table and referee that are connected with a given match. It should be noted that update logic is defined in *Apex*. However, before a record of *Match*


Buttons, Links, and Actions

Action	Label	Name	Description	Type	Content Source	Icon	Overridden
Edit	Accept	Accept			Standard Salesforce.com Page		☐
Edit | Del	Bracket	Bracket		Detail Page Button	Visualforce Page		☐
Edit	Clone	Clone			Standard Salesforce.com Page		☐
Edit	Delete	Delete			Standard Salesforce.com Page		☐
Edit	Edit	Edit			New Tournament (Visualforce Page)		✓
Edit | Del	Entries	Entries		Detail Page Button	Visualforce Page		☐
Edit	List	List			Standard Salesforce.com Page		☐
Edit | Del	Mass Email	MassEmail		Detail Page Button	Visualforce Page		☐
Edit | Del	Matches	Matches		Detail Page Button	Visualforce Page		☐
Edit	New	New			New Tournament (Visualforce Page)		✓
Edit | Del	Results	Results		Detail Page Button	Visualforce Page		☐
Edit	Tournaments Tab	Tab			Standard Salesforce.com Page		☐
Edit	View	View			Standard Salesforce.com Page		☐

Fig. 11 *Tournament* object in *Top 16 Manager*

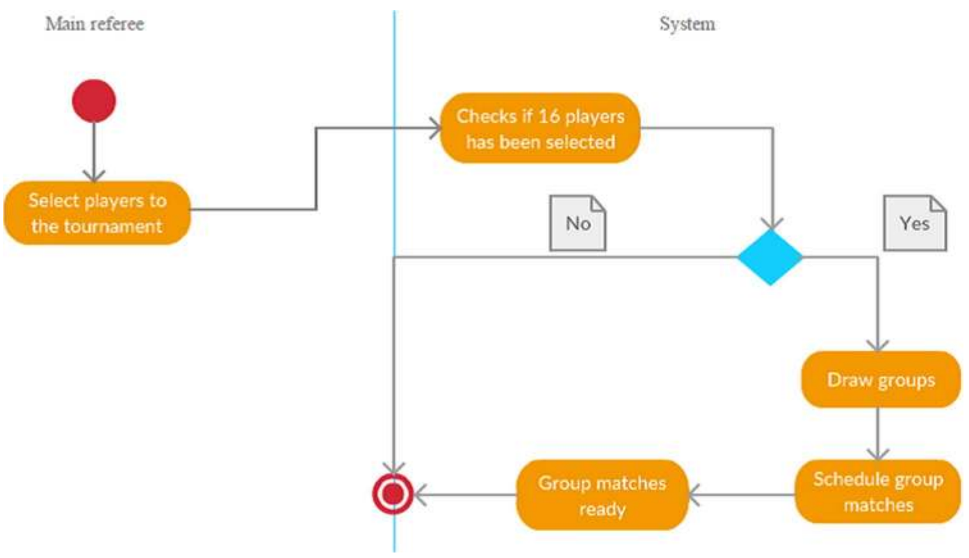


Fig. 12 Activity diagram of enrolling players to the tournament in Top 16 Manager

object is updated, it is checked with the use of a few *Validation Rules* set on this object. If the criteria defined in *the Validation Rules* are satisfied, then the system creates the lists of matches' records, according to the groups in which they take place. It also creates four lists of *Tournament Appearances* object, to calculate the table of each group, since this object stores information about the points gained in the group stage. The system creates a map, where the key is the player's *ID*, and the value is his/her appearance. When the main referee updates matches' results, the procedure goes through matches on the interface, where there is information about the player, and then comparing which of the players has won more frames. Then, it retrieves player's appearance from the map and updates this appearance; accordingly, points are awarded to the winner. Consequently, the group tables are updated. The system queries for the appearances in each group and sorts lists with players by points, the difference between frames won and lost, and frames either won or lost. Figure 13 illustrates the activity diagram of updating group matches.

An example of usage of administration tools is the functionality that sends.

An e-mail, set by the referee on the *Visualforce* interface, is sent to both players after the match time is one example of the administration tools that can provide. The view of this process is depicted in Fig. 14.

As soon as the group stage of the tournament is over, the main referee can generate a bracket for the knock-out phase. This demonstrates the functionalities that are invoked from another *Visualforce* interface and has the logic defined in *Apex* controllers. The custom controller assigns seeding to players

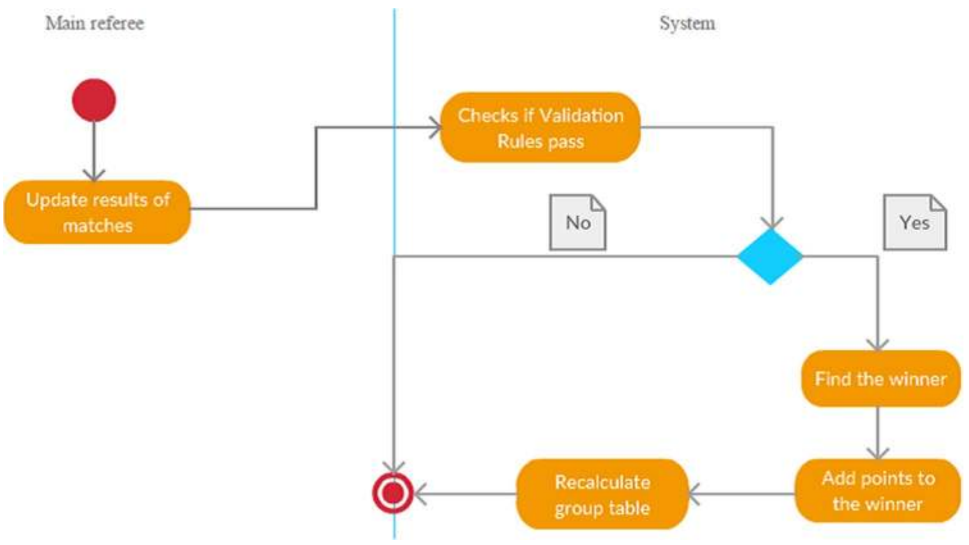


Fig. 13 Activity diagram of group matches updating in Top 16 Manager

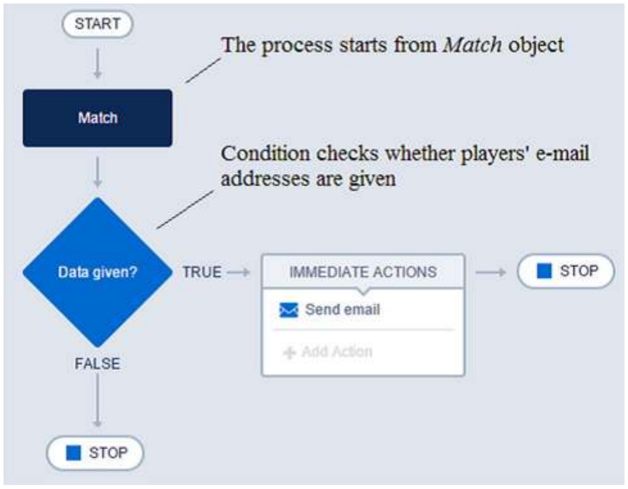

START
The process starts from *Match* object
Match
Condition checks whether players' e-mail addresses are given
Data given?
TRUE
IMMEDIATE ACTIONS
Send email
Add Action
STOP
FALSE
STOP

Fig. 14 The process of sending an e-mail to players with information about next match time

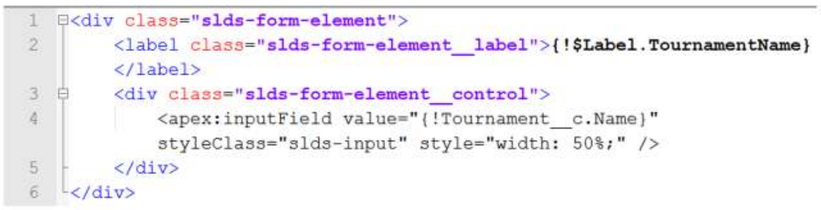
according to their performance in the first phase based on the points they have collected, and the difference between frames won and lost and a higher number of frames won. If all the parameters are equal, then the draw decides which of the players is seeded higher. The system dedicated for this process queries for the appearances of the given tournament and sorts them in the same way as in the group stage. Then, it inserts it into the quarterfinal matches database. The player that is seeded with number 1 competes with the player with number 8. The order is as follows: 2 vs. 7, 3 vs. 6 and 4 vs. 5. The system also inserts the semifinal and final games but does not assign any players for those matches yet. When the tournament bracket is generated, the main referee performs the same actions as in the group stage. However, at this time, the system forms the map in which the key is the matching symbol. For example, QF1, QF2, SF1, etc., and the value is *Match* record. After each update, conditional statements check the winner of each match and assigns him/her to the match in the next round, i.e., either semifinal or the final.

It is possible for the main referee to send a broadcast email to all players taking part in the given tournament by filling the simple form on another custom interface. When the referee submits it, the system queries for the e-mail field of every player taking part in the given competition. Then, with the use of *Apex Messaging* class, it sends all emails. The last custom interface shows the positions that players took in the current tournament that uses *jQuery* plugin *TableSorter*.

To conclude the analysis of actions performed so far, we can confidently say that it is possible to develop a management system based on the *Salesforce* platform. It is not necessary to be developed for business-oriented purposes, although *Salesforce* is mainly aimed for businesses environments. Moreover, it is possible to develop systems to manage individuals' data, using this platform.

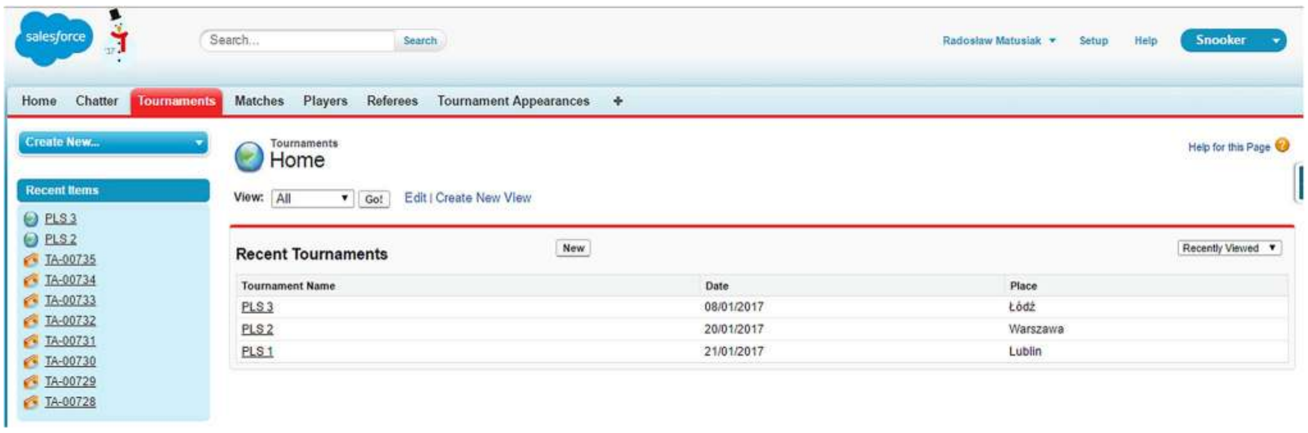
4.5 GUI layer

The implementation of our proposed *Top 16 Manager* system is based on the classic version of Salesforce. To improve the design of the whole system, SLDS is utilized. It is a CSS stylesheet compatible with the lightning experience. What is more, the standard interfaces for creating and editing five custom objects used in *Top 16 Manager* system are overridden by custom ones, which are also styled according to SLDS CSS. To override the standard Salesforce interface, it is necessary to create a custom Visualforce one. For this purpose, a few classes from SLDS stylesheet is applied to make a better look for *Top 16 Manager* system. Listing 1 presents an example for SLDS classes.


```
<div class="slds-form-element">
    <label class="slds-form-element__label">{!$Label.TournamentName}
    </label>
    <div class="slds-form-element__control">
        <apex:inputField value="{!Tournament__c.Name}"
        styleClass="slds-input" style="width: 50%;" />
    </div>
</div>
```

Listing 1. An example of *SLDS* stylesheet classes in overridden standard *Salesforce*

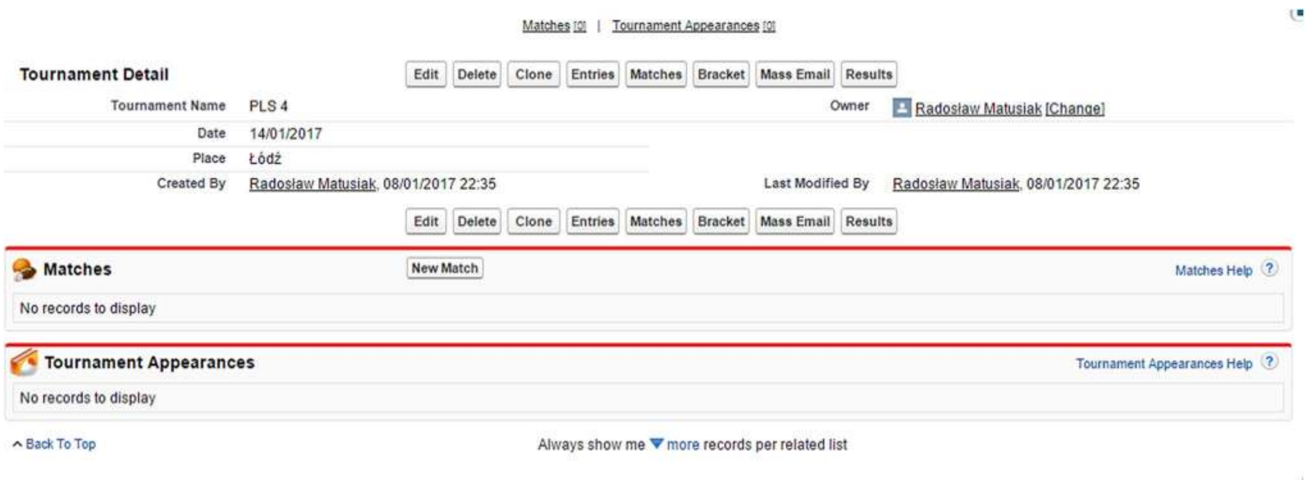
It is possible to merge HTML code with *Visualforce*. Therefore, CSS classes are used in the Listing 1 such as *slds-form-element, slds-form-element__label* or *slds-input*, enabling to style the input fields according to *Lightning* design style wrap the *apex:inputField* element, which in fact it is also rendered to HTML that captures the user input. The example shown in Listing 1 styles only one element from a single interface. According to this approach, all *Visualforce* interface are styled to achieve the *Lightning* look. In order to take advantage of *SLDS* stylesheet, it is loaded from the static resource, where it is uploaded at the beginning. Nevertheless, during the development of *Lightning* styled interface a few problems are encountered such as *SLDS* does not support input fields of type *Lookup*. Moreover, the usage of standard inputs like a date with the intention to display a date picker has damaged the styling of the whole interface.



Recent Tournaments

Tournament Name	Date	Place
PLS 3	08/01/2017	Łódź
PLS 2	20/01/2017	Warszawa
PLS 1	21/01/2017	Lublin

Fig. 15 Main GUI to login to Salesforce platform


Matches [0] | Tournament Appearances [0]
Edit
Delete
Clone
Entries
Matches
Bracket
Mass Email
Results
New Match
Matches Help
Tournament Appearances Help
Back To Top
Always show me more records per related list

Tournament Detail

Tournament Name	PLS 4	Owner	Radosław Matusiak [Change]
Date	14/01/2017		
Place	Łódź		
Created By	Radosław Matusiak, 08/01/2017 22:35	Last Modified By	Radosław Matusiak, 08/01/2017 22:35

Matches

No records to display

Tournament Appearances

No records to display

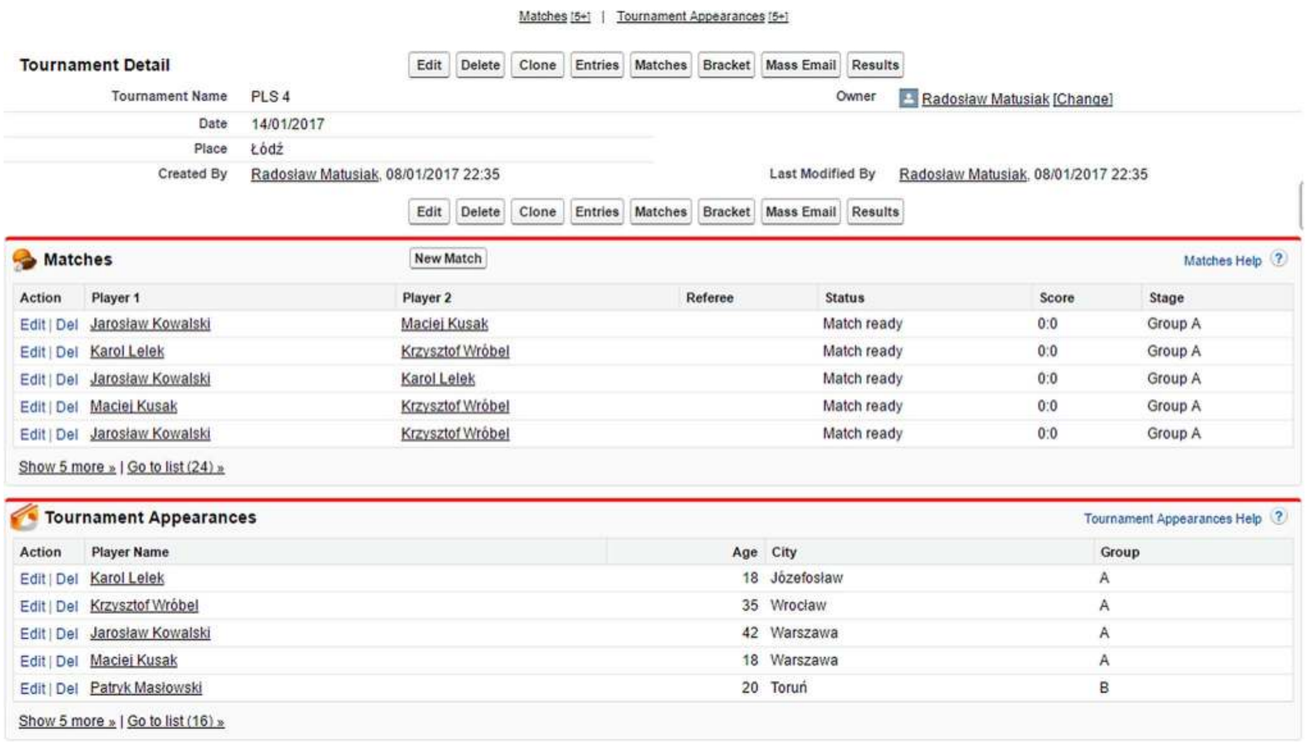
Fig. 16 Tournament interface in Top 16 Manager

Due to these facts, the open source solutions are applied to render the mentioned fields properly. These are a special *Visualforce* component that displays *Lightning* styled inputs in *Salesforce Classic* (Sultan 2014) and the date picker taking advantage of *Appiphony Lightning JS* library (Youseff et al. 2008). Unfortunately, it was not possible to write *New* or *Edit* interface for *Tournament Appearance* object, since *Visualforce* component used in the system supports inputs of type *Lookup* but does not support inputs of type *Master–Detail*.

5 Top 16 Manager in action

This section demonstrates the practical functionalities of the proposed system in Salesforce platform. It is a framework for main referees of the *TOP 16* tournaments. It is presented from the perspective of the system administrator's account. When a user (not a system administrator) successfully login to the platform, he/she can see a panel with tabs at the top of the screen. By default, the tab to which the user is redirected is Tournaments tab as it is the main object in the system. The other tabs concern another custom object presented in the system, including Matches, Players, Referees, and Tournament Appearances. There are also two more default tabs Home and Chatter. They are in the platform to allow users saving some notes, marking important dates in the calendar and for interaction with other users as Chatter that can be described as a company intranet. However, the use of these two tabs is neglected in snooker management system as there is only one user, i.e., the main referee created as an end user and a system administrator. The min GUI promoted to the user is depicted in Fig. 15.

This interface allows a user to select a particular tournament and view its details. It is also possible to expand the *View* list and choose the list view, which displays all records


Matches [5+] | Tournament Appearances [5+]

Tournament Detail

Tournament Name	PLS 4	Owner	Radosław Matusiak [Change]
Date	14/01/2017		
Place	Łódź		
Created By	Radosław Matusiak, 08/01/2017 22:35	Last Modified By	Radosław Matusiak, 08/01/2017 22:35

Matches

Action	Player 1	Player 2	Referee	Status	Score	Stage
Edit | Del	Jarosław Kowalski	Maciej Kusak		Match ready	0:0	Group A
Edit | Del	Karol Lelek	Krzysztof Wróbel		Match ready	0:0	Group A
Edit | Del	Jarosław Kowalski	Karol Lelek		Match ready	0:0	Group A
Edit | Del	Maciej Kusak	Krzysztof Wróbel		Match ready	0:0	Group A
Edit | Del	Jarosław Kowalski	Krzysztof Wróbel		Match ready	0:0	Group A

Show 5 more » | Go to list (24) »

Tournament Appearances

Action	Player Name	Age	City	Group
Edit | Del	Karol Lelek	18	Józefosław	A
Edit | Del	Krzysztof Wróbel	35	Wrocław	A
Edit | Del	Jarosław Kowalski	42	Warszawa	A
Edit | Del	Maciej Kusak	18	Warszawa	A
Edit | Del	Patryk Masłowski	20	Toruń	B

Show 5 more » | Go to list (16) »

Fig. 17 An updated view of tournament details in "Top 16 Manager"

according to the given conditions. After a successful creation of Tournament record, a user is redirected to record's details. It contains all related lists of Matches and Tournament Appearances that are connected with a new tournament. Tournament detail interface has buttons under which there are appropriate functionalities, including: Entries, Matches, Bracket, Mass Email, Results. Figure 16 shows Tournament record interface.

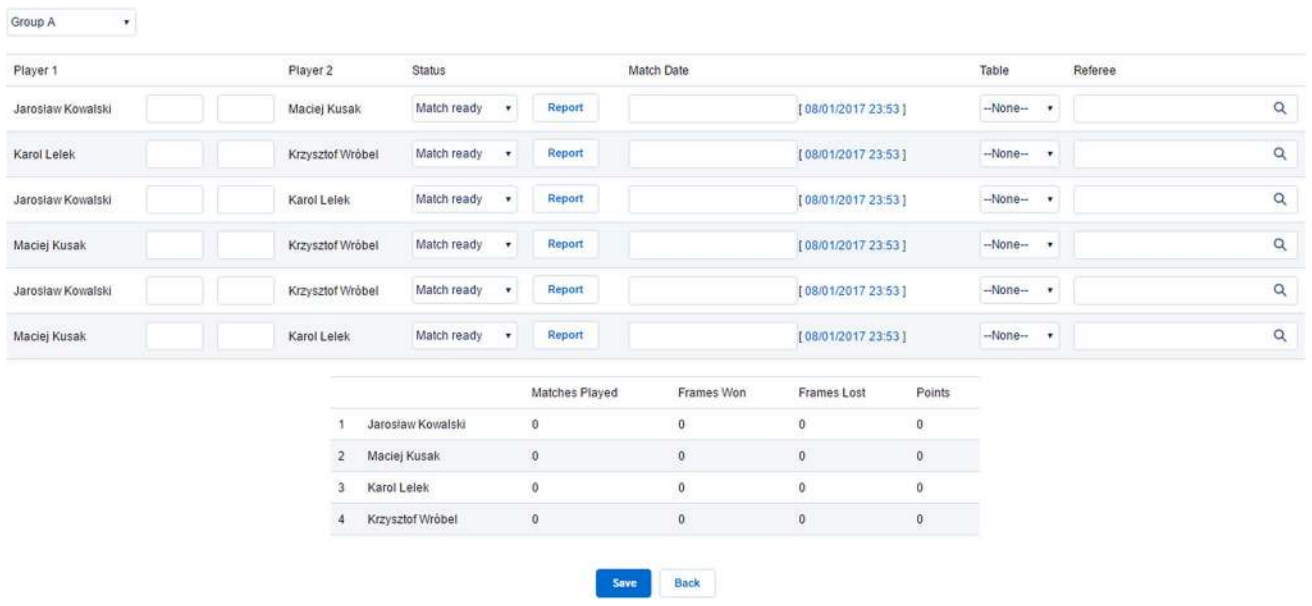
The first step in a tournament management process is to enrolling players to the competition. In our current implementations, there are 20 players, but there might be a necessity in the future for the main referee to add another player. The main referee must tick 16 players that are going to take part in the competition. An updated view of the interface is presented in Fig. 17.

Figure 18 illustrates an interface that shows the desired information about the *Matches*. Through this interface, the main referee can update matches' details.

Match management interface allows updating the current situation in each group. Through this interface, a user not only adds the scores of the finished match, but also update its status (Waiting, Match Ready, In Progress, Finished). A user also is able to generate a report in PDF format and assign a time to each match. Every time the score is added, the information is updated accordingly. The reschedule is constructed even in case of inputting a wrong score. The proposed system does not allow assigning *In Progress* status without a referee assigns it to the match. It also prevents unexpected inputs of score to *TOP 16* tournament. Tie score is not allowed. The winner must have 4 frames won. Also, it is not possible to set two matches with status *In Progress* at the same table. All the aforementioned situations are handled; and the user is prompted with a message on the top of the interface. Every time the main referee updates the time of a single match, the system generates an email to both players with information about match time. It is also possible to send a broadcast email to all players that take part in the given competition.

The last stage of the tournament management process is an update of positions that players took in the competition. Under *Results* button on the *Tournament* details interface, there is a custom interface, which displays the final classification of the tournament, as shown in Fig. 19.

To sum up, all the processes that the main referee is interested in are five custom objects defined especially for the purposes of the required functionalities. Initially, the process starts from either adding a new tournament or finding a proper one. Then, invoking management action from a few custom Visualforce interfaces that are provided on the tournament details interface. A user can also view the data about players or referees. On the related lists of those objects, there is information about achievements or performances. For example, players have their positions taken in each tournament, and referees have listed matches that they



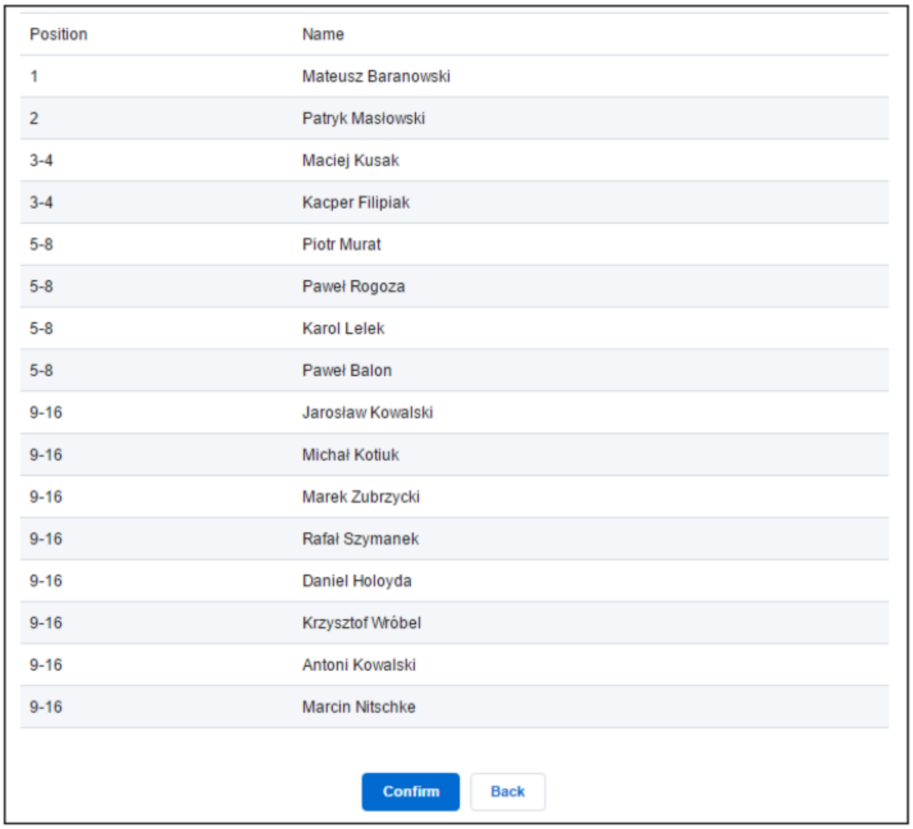
Group A

Player 1			Player 2	Status		Match Date		Table	Referee
Jarosław Kowalski			Maciej Kusak	Match ready	Report		[08/01/2017 23:53]	--None--	
Karol Lelek			Krzysztof Wróbel	Match ready	Report		[08/01/2017 23:53]	--None--	
Jarosław Kowalski			Karol Lelek	Match ready	Report		[08/01/2017 23:53]	--None--	
Maciej Kusak			Krzysztof Wróbel	Match ready	Report		[08/01/2017 23:53]	--None--	
Jarosław Kowalski			Krzysztof Wróbel	Match ready	Report		[08/01/2017 23:53]	--None--	
Maciej Kusak			Karol Lelek	Match ready	Report		[08/01/2017 23:53]	--None--	

		Matches Played	Frames Won	Frames Lost	Points
1	Jarosław Kowalski	0	0	0	0
2	Maciej Kusak	0	0	0	0
3	Karol Lelek	0	0	0	0
4	Krzysztof Wróbel	0	0	0	0

Fig. 18 Match management interface in Top 16 Manager

Fig. 19 Tournament results interface *Top 16 Manager*



Position	Name
1	Mateusz Baranowski
2	Patryk Masłowski
3-4	Maciej Kusak
3-4	Kacper Filipiak
5-8	Piotr Murat
5-8	Paweł Rogoza
5-8	Karol Lelek
5-8	Paweł Balon
9-16	Jarosław Kowalski
9-16	Michał Kotiuk
9-16	Marek Zubrzycki
9-16	Rafał Szymanek
9-16	Daniel Holoyda
9-16	Krzysztof Wróbel
9-16	Antoni Kowalski
9-16	Marcin Nitschke

conducted during *TOP 16* tournaments. User are informed in case incorrect input data.

6 Conclusions and future work

In this paper, we examined the *Salesforce* platform capabilities within the context of the *real-time service systems* and *cloud computing*. The research work performed in this paper proved that the combination of *real-time systems* and *cloud computing* allows building efficient systems that could be used for different purposes. Additionally, the *Salesforce* platform enables the creation of not only the *event-triggered real-time systems* with the use of *Apex Triggers, workflows* and *Process Builder* but also *time-triggered* systems. This is because it is possible to define the time-dependent actions using *Process Builder*. We also developed a *Salesforce* system, i.e., *Top 16 Manager* which aims to help managing the processes of the series of *TOP 16* snooker tournaments held by *Polish Billiards and Snooker Association*. The results showed that the *Salesforce* platform is an environment where it is possible to develop a *soft real-time system* as *private cloud*. The developed system is a combination of *Software as a Service*, from the perspective of the end user, and *Platform as a Service*, from the perspective of the developer. It should be noted that *Top 16 Manager* significantly improves managing of *TOP 16* snooker tournaments that comprises some functionality that is not available for use by main referees. In the future, we plan to make it generic and usable for other types of snooker tournaments and taking advantage of more *Salesforce* functionalities such as reports, dashboards or workflows. We also plan to make *Top 16 Manager* as a component of a larger system dedicated to snooker federations from different countries where snooker competitions take place.

References

Angeles S (2013) 8 reasons to fear cloud computing. In: Bus. News Dly. http://www.businessnewsdaily.com/5215-dangers-cloud-computing.html. Accessed 30 Oct 2017

Barry DK (2003) Web services, service-oriented architectures, and cloud computing. Elsevier, New York

Benioff M, Adler C (2009) Behind the cloud: the untold story of how Salesforce. com went from idea to billion-dollar company-and revolutionized an industry. Wiley, New York

Chen L, Fallmann S, López-de-Ipiña D et al (2018) Context, intelligence and interactions for personalized systems. J Ambient Intell Humaniz Comput 9:1557–1559. https://doi.org/10.1007/s12652-018-0985-y

Computer Security Division ITL Publications|CSRC (2018) https://csrc.nist.gov/publications. Accessed 30 Nov 2018

CRM Software & Cloud Computing Solutions (2018) https://www.salesforce.com/eu/. Accessed 30 Nov 2018

Cusumano M (2010) Cloud computing and SaaS as new computing platforms. Commun ACM 53:27–29

De la Prieta F, Bajo J, Rodríguez S, Corchado JM (2017) MAS-based self-adaptive architecture for controlling and monitoring cloud platforms. J Ambient Intell Humaniz Comput 8:213–221. https://doi.org/10.1007/s12652-016-0434-8

Gomaa H (1993) Software design methods for concurrent and real-time systems, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston

Jin H, Ibrahim S, Bell T, Gao W, Huang D, Wu S (2010) Cloud types and services. In: Furht B, Escalante A (eds) Handbook of cloud computing. Springer, Boston, MA, pp 335–355

Klein M, Ralya T, Pollak B et al (2012) A practitioner's handbook for real-time analysis: guide to rate monotonic analysis for real-time systems. Springer, Berlin

Kopetz H (2011) Real-time systems: design principles for distributed embedded applications, 2nd edn. Springer, Berlin

Krishna CM (2001) Real-time systems. In: Webster JG (ed) Wiley encyclopedia of electrical and electronics engineering. Wiley, New York

Laplante PA (1994) A real-time image processing language?. In: Halang WA, Stoyenko AD (eds) Real time computing. NATO ASI series (Series F: Computer and Systems Sciences), vol 127. Springer, Berlin, Heidelberg

Laplante PA, Ovaska SJ (2011) Real-time systems design and analysis: tools for the practitioner. Wiley, New York

Leavitt N (2009) Is cloud computing really ready for prime time. Growth 27:15–20

Liu F, Narayanan A, Bai Q (2000) Real-time systems

Mattai J (1995) Real-time systems: specification, verification, and analysis. Prentice Hall PTR, Upper Saddle River

Mell P, Grance T (2009) Effectively and securely using the cloud computing paradigm. NIST Inf Technol Lab 2:304–311

Mell PM, Grance T (2011) The NIST definition of cloud computing. Spec Publ NIST SP-800-145

Poniszewska-Maranda A, Matusiak R, Kryvinska N (2017) Use of Salesforce platform for building real-time service systems in cloud. In: 2017 IEEE international conference on services computing (SCC). pp 491–494

Qi J, Xu B, Xue Y et al (2018) Knowledge based differential evolution for cloud computing service composition. J Ambient Intell Humaniz Comput 9:565–574. https://doi.org/10.1007/s12652-016-0445-5

Sadiku MN, Musa SM, Momoh OD (2014) Cloud computing: opportunities and challenges. IEEE Potentials 33:34–36

Salesforce Developers|API Documentation, Developer Forums & More (2017) https://developer.salesforce.com/. Accessed 30 Oct 2017

Schoch DJ, Laplante PA (1995) A real-time systems context for the framework for information systems architecture. IBM Syst J 34(1):20–38

Shin KG, Ramanathan P (1994) Real-time computing: a new discipline of computer science and engineering. Proc IEEE 82:6–24. https://doi.org/10.1109/5.259423

Sriram I, Khajeh-Hosseini A (2010) Research agenda in cloud technologies. arXiv:10013259Cs

Sultan N (2014) Servitization of the IT industry: the cloud phenomenon. Strateg Change 23:375–388. https://doi.org/10.1002/jsc.1983

Tang L, Dong J, Zhao Y, Zhang L-J (2010) Enterprise cloud service architecture. In: IEEE 3rd international conference on cloud computing (CLOUD), 2010, pp 27–34

Trailhead|The fun way to learn Salesforce (2017) https://trailhead.salesforce.com/en. Accessed 30 Oct 2017

Youseff L, Butrico M, Da Silva D (2008) Toward a unified ontology of cloud computing. In: IEEE grid computing environments workshop, 2008. GCE'08, pp 1–10