Ammar Fares Daham

Free Spins Giveaway

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor’s Thesis

1 March 2021

Abstract

Author:	Ammar Fares Daham
Title:	Free Spins Giveaway
Number of Pages:	36 pages
Date:	15 March 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Software Engineering
Supervisors:	Janne Salonen, Director of expertise

The engineering work aims to develop a promotional service for Online casinos and IGaming platforms to give free game rounds within a configured promotional campaign to attract players.

The work started with discussions and planning on structuring this tool effectively, considering it is integration with our current services to ensure compatibility. various structures were proposed, two of those structures were to create this tool within our game system. However, the downside if this tool requires maintains or update it will potentially disrupt the entire system.

Finally, the decision was made to develop the tool as a separate project outside the game system hosted on it is own server with minimal touch to the game system if necessary.

Keywords: Online casino, IGaming, Java, Java servlet, Java web application, Apache tomcat, Postgresql, Hibernate ORM

Tiivistelmä

Tekijä:	Ammar Fares Daham
Otsikko:	Insinöörityön otsikko
Sivumäärä:	36 sivua
Aika:	25.3.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tietotekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Janne Salonen, Osaamisaluejohtaja

Tämän insinöörityön tavoitteena on kehittää online-kasinoille ja IGaming-alustoille promopalvelu, joka tarjoaa ilmaisia pelikierroksia konfiguroidussa promootiokampanjassa pelaajien houkuttelemiseksi.

Työ aloitettiin keskusteluilla ja suunnittelulla tämän työkalun tehokkaasta jäsentämisestä ottaen huomioon, että se on integrointi nykyisten palveluidemme kanssa yhteensopivuuden varmistamiseksi. erilaisia rakenteita ehdotettiin, kaksi näistä rakenteista luovat tämän työkalun pelijärjestelmäämme. Kuitenkin haittapuoli, jos tämä työkalu vaatii ylläpitoa tai päivitystä, se saattaa häiritä koko järjestelmää.

Lopuksi päätettiin kehittää työkalu erillisenä projektina pelijärjestelmän ulkopuolella, sillä se isännöi omaa palvelinta, jossa pelijärjestelmään on tarvittaessa vähän kosketusta.

Avainsanat: Online kasino, IGaming, Java, Java servlet, Java web application, Apache tomcat, Postgresql, Hibernate ORM

Contents

List of Abbreviations

SEO:	Search Engine Optimization
FSCAPI:	Free Spins Campaign API
JVM:	Java Virtual Machine
Java SE:	Java Standard Edition
JSP:	Java Server Page
HTML:	Hyber Text Markup Language
DBMS:	Database Management System. Software for maintaining, querying, and updating data and metadata in a database.
ORM:	Object-relational Mapping. The set of rules for mapping objects in a programming language to records in a relational database, and vice versa.
JDBC:	Java Database Connectivity. A standard Java API for connecting to relational databases.
SQL:	Structured Query Language.
EDR:	Entity Relationship Diagram. A type of flowchart to display the relation between entities.
HTTP:	HyperText Transfer Protocol.
RNG:	Random Number Generator.
JUnit:	Java Unit Testing.

1 Introduction

The main objective of this project is to create a promotional tool called the 'Free Spins giveaway' for Airdice Oy operators. This tool is designed to offer free spin drops to players engaged in Airdice games, with a specific focus on enhancing the overall gaming experience for online casino players. The tool will be working by a pre-configured campaign been configured by Airdice team for their operatores in Malta and South America. The goal is to establish a lasting connection between Airdice Oy and its online casino community in these regions, ultimately leading to increased player retention, revenue growth, and improved market competitiveness.

Our comprehensive exploration will delve into the development phases of this promotional tool, encompassing conceptualization, design, and implementation. Through a seamless integration of cutting-edge technology, our objective is to create a promotional solution that will captivate and engages Airdice's customers.

Commonly employed by online casinos and iGaming platforms, promotional tools serve as effective mechanisms to capture the attention of potential players and enhance the overall appeal of the casino's offerings. These strategies, encompassing a compelling blend of incentives, bonuses, and captivating marketing campaigns, are formulated to promote enthusiasm, inspire involvement, and cultivate a lasting bond between the player and the platform.

In summary, the successful deployment of the "Free Spins giveaway" tool is poised to revolutionize the dynamics of player engagement, financial performance, and market competitiveness for Airdice Oy and its operators. Through this strategic initiative, Airdice aspires to not only retain and attract players but also to thrive as a trailblazer in the competitive landscape of online casinos.

2 Efficient Marketing techniques

In our days the business of iGaming is growing and being extremally competitive market, simply offering a high-quality games or services is no longer sufficient to build successful wealthy enterprise, to thrive in this industry, businesses must leverage efficient marketing techniques, including competitive SEO techniques, Optimize landing pages, promotions and bonuses, with these powerful marketing techniques almost every online casino can become an attractive destination that makes players engage permanently. (Manuela Willbold.)

2.1 Competitive SEO techniques

In the digital world, being easy to find online is super important for all businesses, and one of those businesses is iGaming. It is all about how quickly people can discover your platform on the internet. To show up high in google search results we must use tools called Search Engine Optimization (SEO).

For example, if someone looks up "online casino bonuses," you want your bonus offer to be one of the first things they see. This is even more important if your brand isn't super well-known yet. People who are new to iGaming might start by searching for bonuses.

So, by using specific and easy-to-find words when talking about your stuff online, you make sure that more people can discover your website and what you offer. That's how you make your business more visible online! (Manuela Willbold.)

2.2 Optimize landing page

To increase their online visibility, companies not only invest in premium Search Engine Optimization (SEO) tools but also create special web pages for their important services and products. These pages include specific keyword phrases related to what they offer and eye-catching headlines, captions, and high-quality images or videos. (Manuela Willbold.)

To reach a broader audience, companies also use social media marketing. Since many people use social media platforms like Facebook, Twitter, and Instagram, having accounts and running ads there can help businesses attract new customers. Using the right keywords in their social media posts is crucial to get noticed online. (Manuela Willbold.)

2.3 Promotions and Bonuses

Promotions and bonuses are the strategies and method that online casinos and gaming platforms utilize to engage and reward their players, simultaneously fostering captivating experiences for players while generating profits for the operators.
In our daily lives, it's a common desire to receive bonuses and promotions from businesses we patronize, whether it's a supermarket, a restaurant, or any other service provider. These incentives often make us feel valued as customers and can significantly influence our decisions when choosing where to shop or dine. Similarly, in the world of online casinos, players are also on the lookout for promotions and bonuses that enhance their gaming experience and motivate them to keep playing. (Manuela Willbold.)

The appeal of bonuses and promotions for online casinos is undeniable:

2.3.1 Enhancing customer loyalty

online casino players are drawn to casinos that regularly offer promotions like free spins, deposit bonuses, or cashback rewards. These incentives not only make the gaming experience more enjoyable but also encourage players to stay and wager more within that casino.

2.3.2 Encouraging Engagement

online casinos employ time-limited promotions to stimulate player activity. For instance, a weekend tournament or campaign with higher prize pools or a special event with extra bonuses can motivate players to participate more actively during those specific periods.

Let's pause for a moment and shift our focus to an exciting promotional tool “free spins”. Online casinos leverage this tool to not only promote our games but also to reward and entice their players. Free spins are a powerful incentive that not only enhances the gaming experience but also serves as a key driver for increased player engagement.

Now, let's delve deeper into the significance of free spins and their pivotal role within the dynamic of marketing promotion tool for online casinos.

3 Free spins promotion tools

Free spins serve as a dynamic marketing tool and an engaging feature for online casinos and gaming platforms. This promotional offering involves granting players a specific number of complimentary attempts as part of a welcome bonus, or during promotional campaign allowing them to enjoy a game round without having to wager their own money. Free spins come in two primary forms: no-deposit bonuses and inclusion within deposit bonus packages, and in this document, we will focus on the no-deposit bonus variant, where we have identified and restricted specific type called free spins giveaway.

4 Planning phase

The planning phase started with drawing a diagram, describing the existing servers along with the proposed addition the promo server. Promo server constitutes an important role in configuring, reading free spin campaigns, and deciding based on timestamp to award free spin drops. The process begins when a player launches a game session, the client dispatches a request to the promo server asking if there is an active campaign. The promo server sends back response indicating the presence or absence of an active campaign. If a campaign is active, the client receives detailed information about it, enabling the player to initiate game session.

The promo server receives replicated data after each finished game round from the central server and make a timestamp-based decision to give free spin drops and tell the FSCAPI to officially award these promotional wins. Afterwards, the client queries the promo server to check for promotional wins, receiving a detailed response. This seamless interaction between the client, promo server, FSCAPI and central server indicates the efficient management and communication of information regarding ongoing promotional activities. Notably, the promo server maintains its own database to store promotional campaigns and track the limited drops intended to be distributed during each campaign.

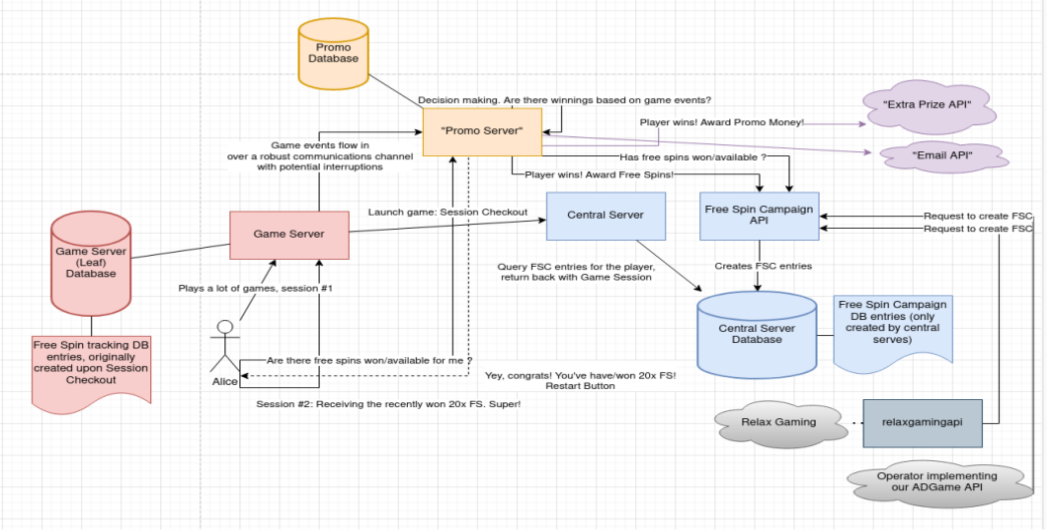

Promo Database
Decision making. Are there winnings based on game events?
"Promo Server"
Player wins! Award Promo Money!
"Extra Prize API"
"Email API"
Game events flow in over a robust communications channel with potential interruptions
Has free spins won/available ?
Player wins! Award Free Spins!
Launch game: Session Checkout
Game Server
Central Server
Free Spin Campaign API
Request to create FSC
Request to create FSC
Game Server (Leaf) Database
Plays a lot of games, session #1
Query FSC entries for the player, return back with Game Session
Creates FSC entries
Central Server Database
Free Spin Campaign DB entries (only created by central serves)
Free Spin tracking DB entries, originally created upon Session Checkout
Alice
Are there free spins won/available for me ?
Yey, congrats! You've have/won 20x FS! Restart Button
Session #2: Receiving the recently won 20x FS. Super!
Relax Gaming
relaxgamingapi
Operator implementing our ADGame API

Figure 1. Game System

4.1 Promos from server

The Promo server is an important part of the dynamic process, it receives replicated data after each finished game round from the game server. When receiving the data needed from game server, the Promo server takes on a time-based decision-making algorithm to give free spin drops. Afterwards, it initiates a call to the "CreateCampaign" function from FSCAPI to award these free drops. FSCAPI, is one of the existing services in our system that, facilitates communication between the promo server, the central server, and itself. After successfully awarding these drops, FSCAPI sends responses to the central server and to the promo server. These responses guarantee the success or failure of the request. Additionally, the promo server, acknowledges the importance of saving campaign information, persists the unique campaign Id associated with the awarded free spin drops.

This collaborative workflow ensures the seamless execution of promotional awards, with the promo server acting as a central hub for decision-making, while the FSCAPI serves as a bridge, managing the flow of information between different components of the system. This process is illustrated in the "promos_server_pov" diagram.

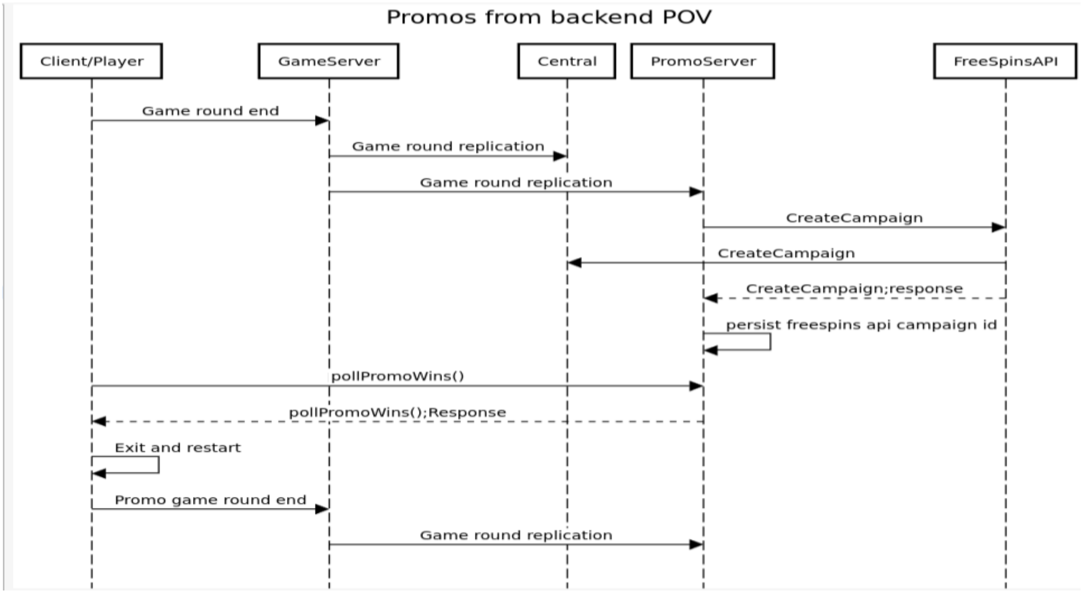


Figure 2. Promo Server Side

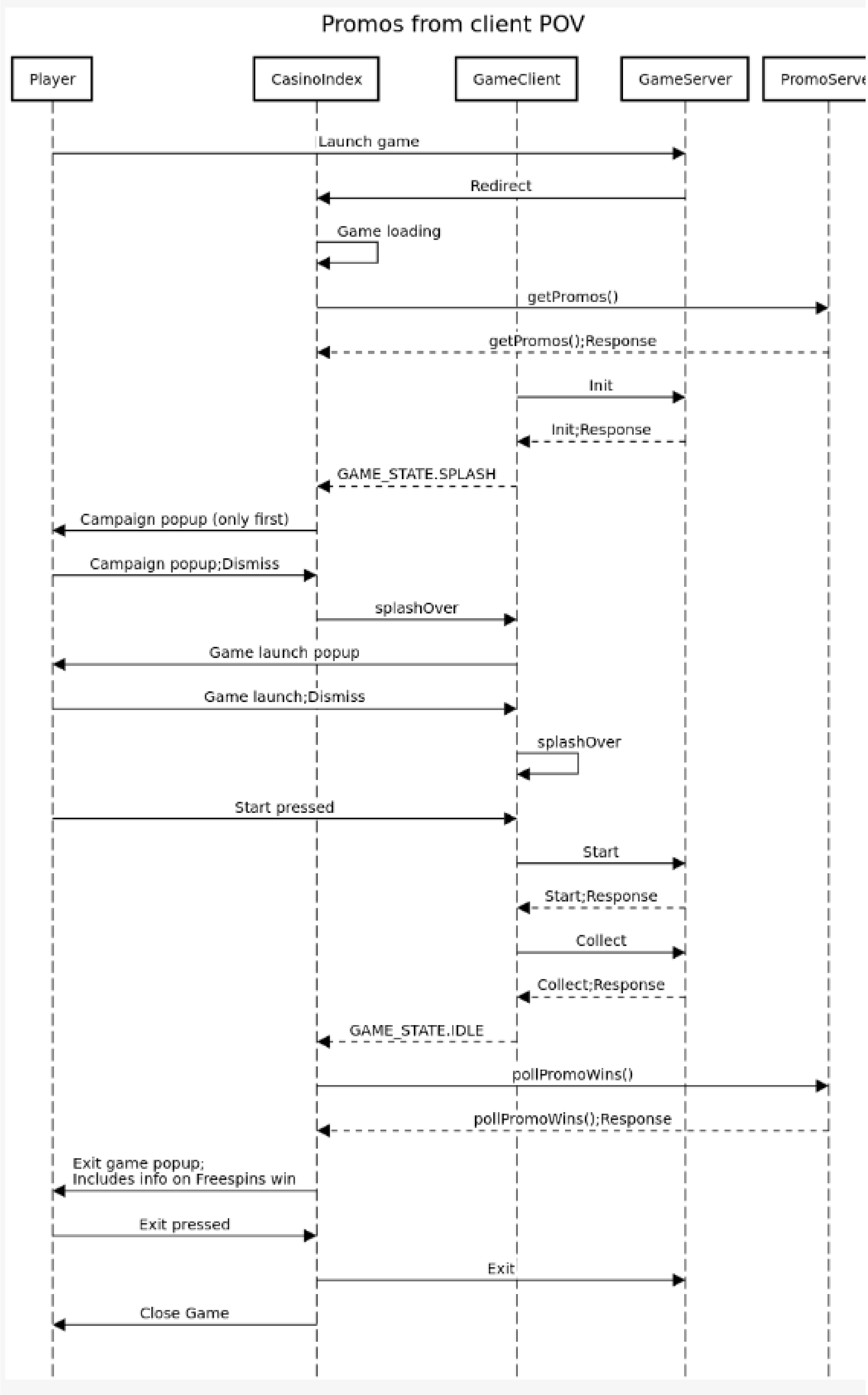
4.2 Promos from client

The promo server's functionality from the client's perception, starts as the player launches the game through the game server, then player gets redirected to the casino index. Afterwards, the game loading, and the client sends a "getPromos" request to the promo server.

When the client receives response, there will information about the existence or non-existence of an active promotional campaign. If there is an active campaign, a campaign popup appears, displaying campaign details to the player. Thereafter, the player starts a game round, and after each collect request, which means the game round has been finished, the client dispatches a "pollPromoWins" request, prompting the promo server to respond by either awarding promo wins or not.

As the promo server grants free spin drops, the client hands over information about the awarded promotional wins and informing the player to exit the game and start new session to access and play the free spin drops. The game is relaunched, sending another "getPromos" request, and upon receiving the response regarding the active campaign, the player gains authorization to play the awarded free spin drops, note that the player prohibited to get a free spin drop based on a free spin game round.

Through the game session, the free spins counter is dynamically updated by the client, decreasing it after each finished free spin game round. After all free spin game rounds are completed, the client hides the free spins counter and displays a popup message saying, "free credits end", providing the players with a summary of their overall winnings. The comprehensive sequence diagram provided below illustrates the interaction between the promo server and the client, detailing each step of the process from game start to the peak of free spin rounds and the display of end outcomes.


Promos from client POV
Player
CasinoIndex
GameClient
GameServer
PromoServe
Launch game
Redirect
Game loading
getPromos()
getPromos();Response
Init
Init;Response
GAME_STATE.SPLASH
Campaign popup (only first)
Campaign popup;Dismiss
splashOver
Game launch popup
Game launch;Dismiss
splashOver
Start pressed
Start
Start;Response
Collect
Collect;Response
GAME_STATE.IDLE
pollPromoWins()
pollPromoWins();Response
Exit game popup;
Includes info on Freespins win
Exit pressed
Exit
Close Game

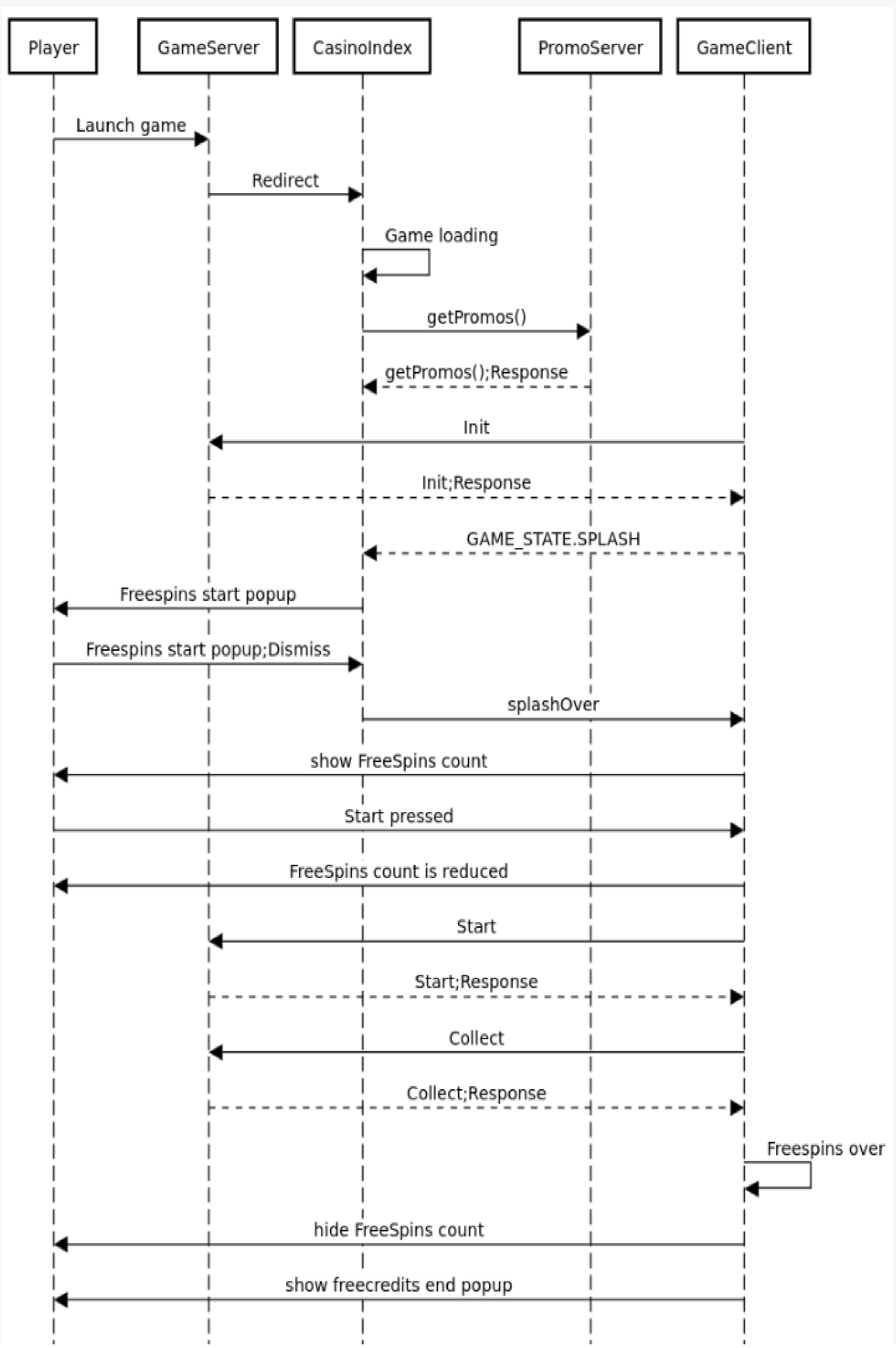
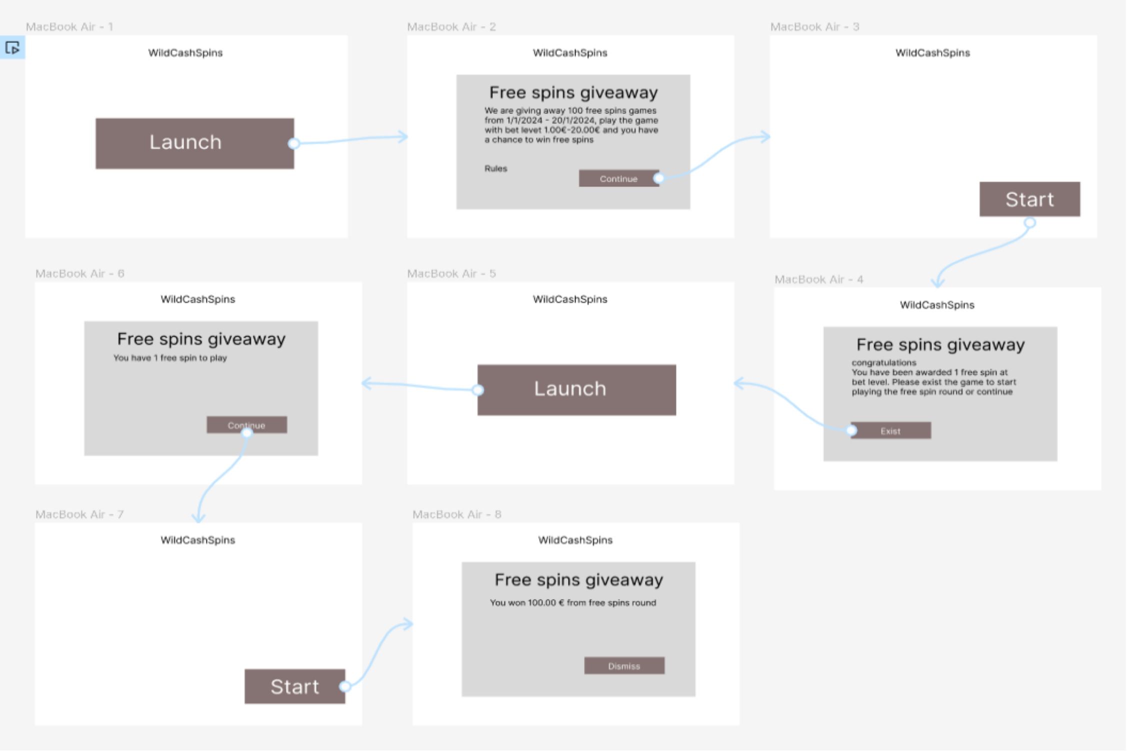


Figure 3. Promo Client Side

In the screen-mock-up provided below we aim to display the interaction in the client throughout the entire gaming session from initiating a game session to awarding free spin drops, afterward playing the awarded free spin drops. This illustration serves to showcase the user journey.



We are giving away 100 free spins games from 1/1/2024 - 20/1/2024, play the game with bet levet 1.00€-20.00€ and you have a chance to win free spins

congratulations
You have been awarded 1 free spin at bet level. Please exist the game to start playing the free spin round or continue

You have 1 free spin to play

You won 100.00 € from free spins round

Figure 4. Promo Screen Mock-Up

In the flowchart provided, each step explained, identifying the areas where popups are triggered. Upon launching any game if a promotional campaign going on, the client will present a notification popup with 'OK' button. Afterward the player presses the button, player start a game round, when the game round finished the client checks there are awarded free spin drops, then prize popup presents with 'Exit' button, prompting the player to reload the game and start new session. Following that the client displays a free spins counter allowing the player to play the free rounds. In the absence of awarded free spins the player continues playing another game round and so on. if no campaign is active no notification popup appears, and the player can play normally assuming there is no promotional campaign going on.

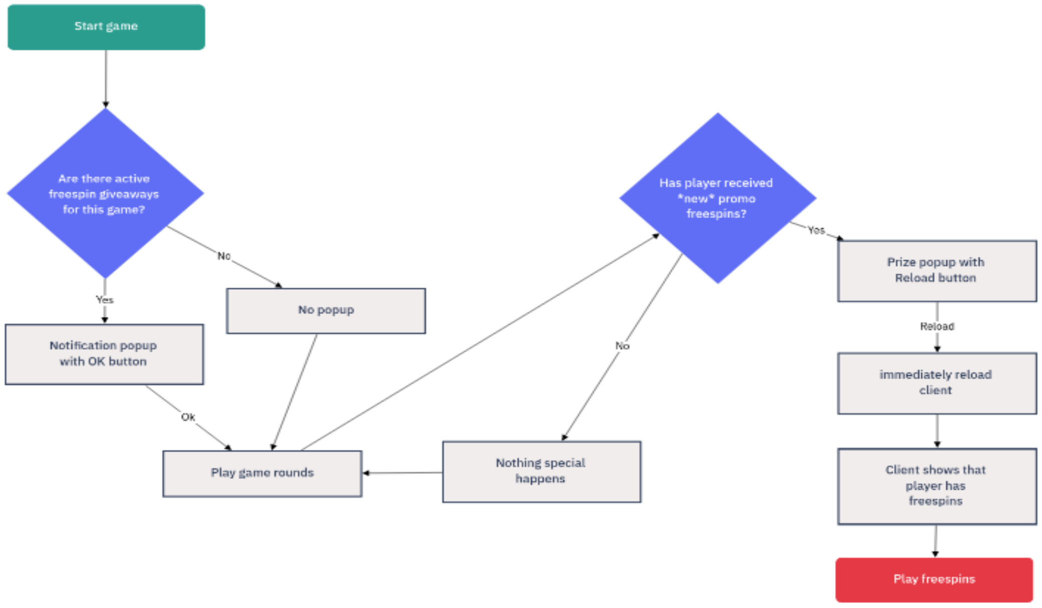

Start game
Are there active freespin giveaways for this game?
No
Yes
Notification popup with OK button
No popup
Ok
Play game rounds
Nothing special happens
No
Has player received *new* promo freespins?
Yes
Prize popup with Reload button
Reload
immediately reload client
Client shows that player has freespins
Play freespins

Figure 5. UX flowchart

5 Core technologies applied in the promo server

The Promo Server is a Gradle web application that utilizes Java Servlet and a variety of advanced technologies to ensure optimal performance, scalability, and user experience. This well-thought-out integration is designed to provide a seamless and efficient platform for users. Promo server utilizes the following cutting-edge development technologies:

- Java 11
- PostgreSQL 14 as a DBMS.
- Hibernate 5.6
- Apache Tomcat 9 server infrastructure.
- JGroups cluster for data replication.
- JUnit.

5.1 Java

Java is an object-oriented programming language and a development platform that runs on billions of devices worldwide, initially backed by Sun Microsystems and currently supported by Oracle, stands out for its open-source nature. Java is considered as a multiplatform, it was branded with a saying "write once, run anywhere" or (WORA) and that explains java code can be written for one platform and easily transferred to another platform without being completely rewritten, the complier creates .class bytecode that can be run on any operating system that has Java virtual machine (JVM) installed. In our software, we specifically employed Java EE with servlet 4.0, which widely used for programming web applications in the Java programming language to ensure seamless compatibility with our existing services, all of which are already integrated with Java 11. (Jaiswal Sonoo)

5.1.1 Java Web Application

Java web application is used to create a dynamic website, and Java language provides this feature through servlet and JSPs, you might be wandering that we can create websites using Hyper Text Markup Language (HTML) which is easier, but that website will be a static, in case if we want our information to be dynamic then, we use Java web application. We have used "Eclipse IDE with java EE developers" for creating a servlet application. (Jaiswal Sonoo)

5.1.2 Java EE

Java Enterprise Edition (EE) platform is a collection of APIs and tools to create web application. Java EE is a specification that application servers and software development environments must follow. Java EE contains several API definitions such as JDBC, RMI, e-mail, JMS, web services, XML, etc. In addition to these, Java EE contains component definitions such as Enterprise JavaBeans, servlet, portlet and JavaServer Pages. (Cheah 2019.)

5.1.3 Servlet

Java servlets are a Java program classes which extend the functionality of servers that runs on web server or application server, it is used to handle requests and send back responses vie Http servlet classes, servlet is a part of Java EE platform. We utilize javax.servlet and javax.servlet.http packages to write the servlet's interfaces and classes, the httpServlet class provides methods like "doGet" and "doPost" for handling HTTP services. Below an example of initializing servlet. (Cheah 2019.)

```
public class PromoServlet extends HttpServlet {

    private PromoDao promoDao;
    private SettingServiceImpl settingService;
    private DbTransactionManager transactionManager;
    private PromoEngine promoEngine;

    @Override
    public void init(ServletConfig config) throws ServletException {

        super.init(config);
        ThreadCreator.getInstance().start();

        log.info("Initializing promoserver v{}");

        jsonMapper.setSerializationInclusion(Include.NON_NULL);
        jsonMapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTI
        ES, false);
        jsonMapper.registerModule(JsonConverters.module());

        settingService = new SettingServiceImpl();
        settingService.startPolling();

        transactionManager = new DbTransactionManager();

        promoDao = new PromoDao(transactionManager);
        promoEngine = new PromoEngine(promoDao, settingService);
        transactionManager.startTransaction();

        // The purpose of this method is to set first award times for all
        // active campaigns, upon each restart of the service.
        try {
            promoEngine.getAllPromos();
        }
        finally {
            transactionManager.endTransaction();
        }
    }
}
```

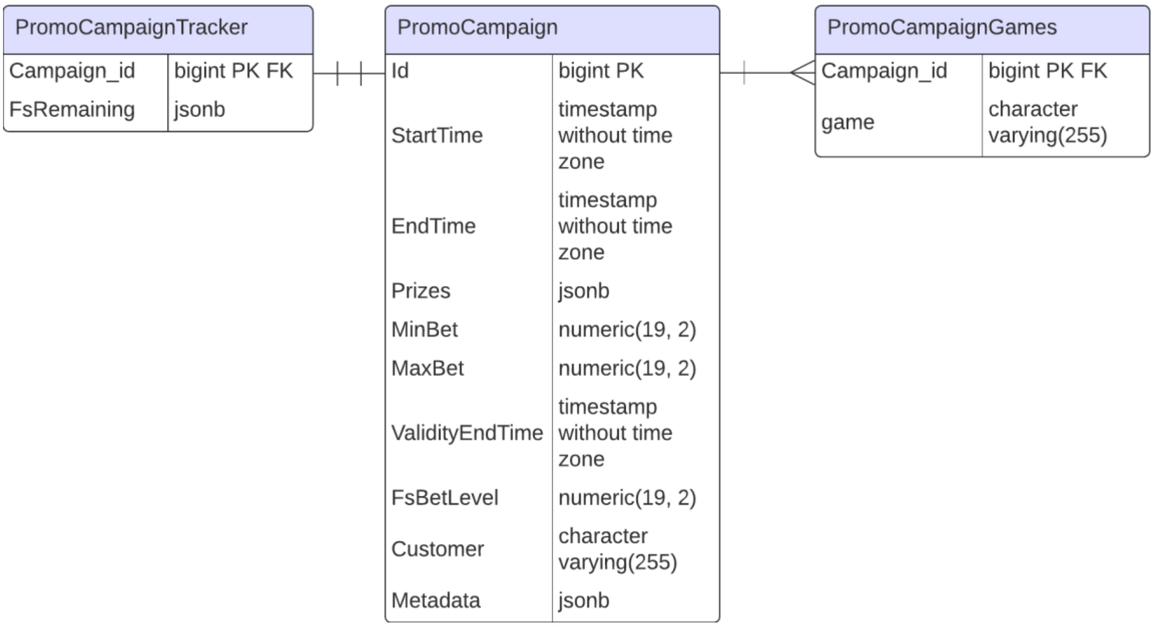
Listing 1. Promo servlet initialization

5.2 PostgreSQL

PostgreSQL is a free, powerful, open-source and progressive object-relational database. It is widely noted for its popularity, scalability, extensibility, robustness, and reliability. Significantly, Promo server utilized of postgresql 14 as it is the default version in Ubuntu 22.4. (What is postgresql.)

The ERD diagram below, describes the promo server entities and illustrates their relationship. There are three entities: PromoCampaign responsible for storing information regarding configured campaigns, PromoCampaignTracker tracks the awarded prizes, and PromoCampaignGames intended to shore the games configured for the campaign.

PromoCampaign shares a one-to-one relation with PromoCampaignTracker, highlighting a direct and singular association. while PromoCampaignGames shares a one-to-many relation with PromoCampaign, implying that multiple games can be configured under a single campaign.



PromoCampaignTracker	
Campaign_id	bigint PK FK
FsRemaining	jsonb

PromoCampaign	
Id	bigint PK
StartTime	timestamp without time zone
EndTime	timestamp without time zone
Prizes	jsonb
MinBet	numeric(19, 2)
MaxBet	numeric(19, 2)
ValidityEndTime	timestamp without time zone
FsBetLevel	numeric(19, 2)
Customer	character varying(255)
Metadata	jsonb

PromoCampaignGames	
Campaign_id	bigint PK FK
game	character varying(255)

Figure 6. Promo ERD

Table 1. PromoCampaign table’s fields in the database.

Field	**Description**
Id	hibernate sequence number defines the campaign identifier
startTime	Defines the start time of the campaign
EndTime	Defines the end time of the campaign
Prizes	Defines the campaign prizes of the campaign
MinBet	Defines the minimum bet level of the campaign
MaxBet	Defines the maximum bet level of the campaign
ValidityEndTime	Defines the expired time of the awarded prizes
FsBetLevet	Defines optional bet level for the awarded prizes
Customer	Defines the customer of the campaign
Metadata	Defines name and rules of the campaign

Table 2. PromoCampaignGames table’s fields in the database.

Field	**Description**
Campaign_id	hibernate sequence number defines the campaign identifier
Game	Defines the games allowed in the campaign

Table 3. PromoCampaignTracker table’s fields in the database.

Field	**Description**
Campaign_id	hibernate sequence number defines the campaign identifier
FsRemaining	Defines the remaining prizes of the campaign

After the database structure have been designed, the database creation phase started by creating the database schema, below is the schema for the Promo Server database.

```
ALTER TABLE ONLY public.promocampaigntracker DROP CONSTRAINT
promocampaigntracker_campaign_id_fkey;

ALTER TABLE ONLY public.promocampaigntracker DROP CONSTRAINT
promocampaigntracker_pkey;

ALTER TABLE ONLY public.promocampaigngames DROP CONSTRAINT
promocampaigngames_campaign_id_fkey;

ALTER TABLE ONLY public.promocampaigngames DROP CONSTRAINT
promocampaigngames_pkey;

ALTER TABLE ONLY public.promocampaign DROP CONSTRAINT promocampaign_pkey;

ALTER TABLE public.promocampaign ALTER COLUMN id DROP DEFAULT;

DROP TABLE public.promocampaigntracker;
DROP TABLE public.promocampaigngames;
DROP TABLE public.promocampaign;
DROP SEQUENCE hibernate_sequence;


CREATE TABLE promocampaign (
     id bigint NOT NULL,
     starttime timestamp without time zone NOT NULL,
     endtime timestamp without time zone NOT NULL,
     prizes jsonb NOT NULL, minbet numeric(19,2),
     maxbet numeric(19,2),
     validityendtime timestamp without time zone NOT NULL,
     fsbetlevel numeric(19,2),
     customer character varying(255) NOT NULL, metadata jsonb NOT NULL
);

CREATE TABLE promocampaigngames (
     campaign_id bigint NOT NULL,
     game character varying(255) NOT NULL
);

CREATE TABLE promocampaigntracker (
     campaign_id bigint NOT NULL,
     fsremaining jsonb NOT NULL
);

ALTER TABLE promocampaign ADD CONSTRAINT promocampaign_pkey PRIMARY KEY (id);

ALTER TABLE promocampaigngames ADD CONSTRAINT promocampaigngames_pkey PRIMARY
KEY (campaign_id, game);

ALTER TABLE promocampaigngames ADD CONSTRAINT
promocampaigngames_campaign_id_fkey FOREIGN KEY (campaign_id) REFERENCES
promocampaign(id);

ALTER TABLE promocampaigntracker ADD CONSTRAINT promocampaigntracker_pkey
PRIMARY KEY (campaign_id);

ALTER TABLE promocampaigntracker ADD CONSTRAINT
promocampaigntracker_campaign_id_fkey FOREIGN KEY (campaign_id) REFERENCES
promocampaign(id);

CREATE SEQUENCE hibernate_sequence AS bigint START WITH 1 INCREMENT BY 1 NO
MINVALUE NO MAXVALUE CACHE 1;
```

Listing 2. Promo server database schema.

5.3 Hibernate

Is a java framework, well known for it is high performance, scalability, reliability, extensibility, and configurability. It is used to simplify the development of java application to interact seamlessly with the database. This open-source Object-Relational Mapping (ORM) tool helps for data creation, manipulation, access, and mapping java objects into database tables, in another word hibernate ORM is concerned to achieve data persistence in java applications. Hibernate utilizing JDBC to establishe a connection with the relational database, which is a lower level requires to write SQL queries and handle the results manually. (Hibernate ORM.)

5.4 Apache Tomcat

Apache tomcat is an open-source implementation of web server and servlet container for java code, which was started in 1998 at microsystems and donated to Apache software foundation in 1999 and it becomes a widely used and trusted platform among java developers. Tomcat offers robust support for Java servlets and Java server pages (JSP), making it a preferred choice for developers. (Tomcat documentation.)

Basically, tomcat provides HTTP web server environment for java code to run as a dynamic and interactive web application. Deploying the application to the server is easy just copy a compressed .war file under CATALINA_BASE/webapps/. below are some of the tomcat directories:

- /bin – startup, shutdown, and other script.
- /conf – configuration files.
- /logs – logs files.
- /webapps – this where webapps go.

Our promo server utilizes tomcat 9 that supports servlet 4.0 to be compatible with other existing services in our game system.

5.5 JGroups

Jgroups is a toolkit for trustworthy sending and receiving messages, it has been written in Java and it is used for creating clusters and it provides a set of Java APIs that simplify communication and coordination between nodes within a cluster, it saves developer's time as there is no need for implementation by the application's developers. (JGroups documentation.)

5.6 JUnit (Java Unit Testing)

JUnit is a free and open-source java framework, it is a unit testing tool that can be used to compare the desired state and the received response. If they are not match, JUnit sends an error. In the provided code source below, we examine some of the test cases for promo server. (JUnit FAQ.)

```
public class PromoEngineTest {
        private PromoEngine promoEngine;
        private PromoDao mockPromoDao;
        private SettingServiceInterface mockSettingService;
        private List<PromoCampaign> mockPromoCampaigns;
        private String customerId = "customer1";
        private String gameId = "game1";
        private List<String> games = new ArrayList<String>();
        private Instant timeNow = Instant.now();

        @Before
        public void setUp() {
           mockPromoDao = mock(PromoDao.class);
           mockSettingService = mock(SettingServiceInterface.class);
           promoEngine = new PromoEngine(mockPromoDao,
           mockSettingService);
           // Mock campaign data
           games.add(gameId);
           Timestamp startTime = Timestamp.valueOf("2024-03-11
           07:00:00");
           Timestamp endTime = Timestamp.valueOf("2024-03-15
           18:00:00");
           Prizes prizes = new Prizes();
           prizes.setPrizesPerDrop(Arrays.asList(10, 20, 30));
           prizes.setDropLimits(Arrays.asList(100, 200, 300));
           Metadata metadata = new Metadata("wide-bet-campaign",
           "http://example.com");
           mockPromoCampaigns = new ArrayList<>();
           mockPromoCampaigns.add(new PromoCampaign(games, startTime,
           endTime, prizes, new BigDecimal("0.10"), new
           BigDecimal("10.00"), customerId, metadata));
           metadata = new Metadata("specific-bet-campaign",
           "http://example.com");
           mockPromoCampaigns.add(new PromoCampaign(games, startTime,
           endTime, prizes, new BigDecimal("1.00"), new
           BigDecimal("1.00"), customerId, metadata));
           metadata = new Metadata("small-bet-campaign",
           "http://example.com");
           mockPromoCampaigns.add(new PromoCampaign(games, startTime,
           endTime, prizes, new BigDecimal("0.10"), new
           BigDecimal("1.00"), customerId, metadata));
           metadata = new Metadata("too-small-bet-campaign",
           "http://example.com");
           mockPromoCampaigns.add(new PromoCampaign(games, startTime,
           endTime, prizes, new BigDecimal("0.10"), new
           BigDecimal("0.99"), customerId, metadata));
           metadata = new Metadata("big-bet-campaign",
           "http://example.com");
           mockPromoCampaigns.add(new PromoCampaign(games, startTime,
           endTime, prizes, new BigDecimal("1.00"), new
           BigDecimal("100.00"), customerId, metadata));
           metadata = new Metadata("too-big-bet-campaign",
           "http://example.com");
           mockPromoCampaigns.add(new PromoCampaign(games, startTime,
           endTime, prizes, new BigDecimal("1.01"), new
           BigDecimal("100.00"), customerId, metadata));
           when(mockPromoDao.getActivePromos(customerId, gameId,
           Timestamp.from(timeNow))).thenReturn(mockPromoCampaigns);
        }

        @Test
        public void testGetActivePromosWithSmallBet() throws
        JsonProcessingException {
           List<PromoCampaign> activePromos =
           promoEngine.getActivePromos(customerId, gameId, timeNow,
           new BigDecimal("1.00"));
           assertEquals(4, activePromos.size()); assertEquals("wide-
           bet-campaign",
           activePromos.get(0).getMetadata().getCampaignName());
           assertEquals("specific-bet-campaign",
           activePromos.get(1).getMetadata().getCampaignName());
           assertEquals("small-bet-campaign",
           activePromos.get(2).getMetadata().getCampaignName());
           assertEquals("big-bet-campaign",
           activePromos.get(3).getMetadata().getCampaignName());
       }

       @Test
       public void testGetActivePromosWithBigBet() throws
        JsonProcessingException {
           List<PromoCampaign> activePromos =
           promoEngine.getActivePromos(customerId, gameId, timeNow,
           new BigDecimal("11.00"));
           assertEquals(2, activePromos.size());
           assertEquals("big-bet-campaign",
           activePromos.get(0).getMetadata().getCampaignName());
           assertEquals("too-big-bet-campaign",
           activePromos.get(1).getMetadata().getCampaignName());
        }
    }
```

Listing 3. Promo server junit test.

6 Results

As an outcome of promo server, we have configured a promotional campaign for a 20-minute duration as displayed in the json object below:

```
{ “id”: 28, “games”: [ “gameId” ], “startTime”: “2024-03-07 14:40:00”,
“endTime”: “2024-03-07 15:00:00”, “prizes”: { “dropLimits”: [ 20, 30, 10 ],
“prizesPerDrop”: [ 2, 3, 1 ] }, “minBet”: 0.25, “maxBet”: 50.00,
“validityEndTime”: “2024-03-10 15:00:00”, “customer”: “customer_test”,
“metadata”: { “campaignName”: “test”, “rulesLink”: “” },
“promoCampaignTracker”: { “campaign_id”: 28, “fsRemaining”: { “dropLimits”: [
20, 30, 10 ], “prizesPerDrop”: [ 2, 3, 1 ] } } }
```

Listing 4. Configured promotional campaign object.

Running promo server on the following campaign would award the allocated prizes and, in the table, below displaying the awarded prizes grounded in a time-based distribution mechanism.

Table 2. Time-based award distribution

Time	Seconds	Prize1	Prize2	Prize3	Interval (s)
14:40:05.48	0		2		
14:40:24.61	20		2		20
14:41:05.87	61			3	41
14:41:25.75	81			3	20
14:41:47.15	102		2		21
14:42:13.52	129		2		27
14:42:35.83	151			3	22
14:42:59.22	174			3	23
14:43:21.18	196			3	22

Time	Seconds	Prize1	Prize2	Prize3	Interval (s)
14:43:58.78	234			3	38
14:44:38.80	274	1			40
14:45:01.70	297		2		23
14:45:22.84	318	1			21
14:45:45.26	340			3	22
14:46:06.40	361	1			21
14:46:47.64	403		2		42
14:47:22.92	438			3	35
14:47:55.89	471		2		33
14:48:33.06	508	1			37
14:49:10.33	545	1			37
14:49:45.81	581	1			36
14:50:09.97	605	1			24
14:50:40.51	636			3	31
14:51:30.48	685			3	49
14:52:43.29	758		2		73
14:53:51.49	826	1			68
14:55:03.85	899	1			73
14:56:16.82	972		2		73
14:57:12.12	1027		2		55

Time	Seconds	Prize1	Prize2	Prize3	Interval (s)
14:58:39.67	1115	1			88
15:00:00Z					

In this promotional campaign we have configured three different prizes, each valued at 10, 20 and 30 drops distributed randomly utilizing random number generator (RNG) java library, meaning that if the prize is the first one a single drop is given, for the second prize two drops is given and for the third prize three drops is given. This procedure will continue until the campaign end time has become to it is end or all the allocated prizes have been distributed. The table below demonstrate the amount of the prizes, minimum, maximum and average interval of the awarded time.

Table 3. Prizes interval.

	Prize 1	Prize 2	Prize 3
Prizes amount (n)	10	20	30
Min interval (s)	20		
Average interval (s)	38.45		
Max interval (s)	88		

The scatter diagram below explains how prizes have been distributed.

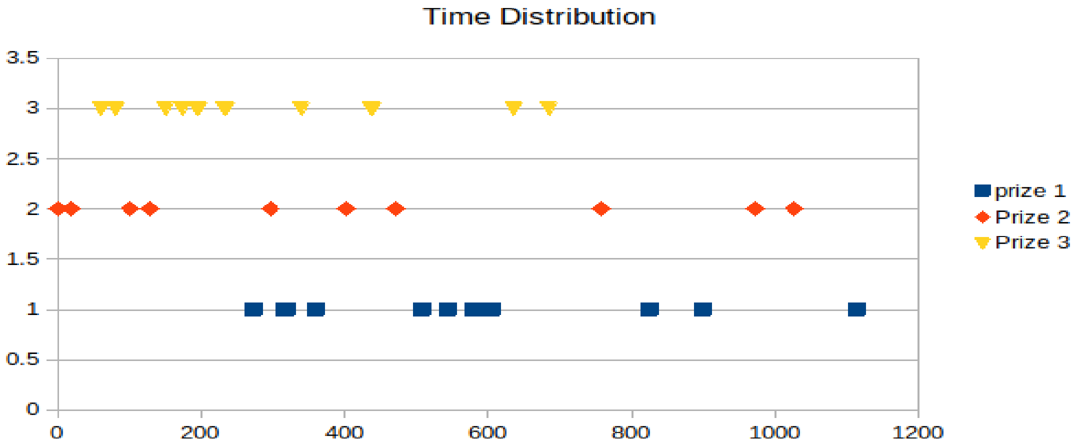


Figure 7. Time-based distribution

7 Summary

The original goal of the thesis was to develop a promo server demo to create campaigns and in those active campaigns decide when to award free game rounds to player based on the timestamp and the quantity of free drops that the casino is willing to distribute.

Designing and planning the software architecture is a complex task, it must align closely with business requirements and the technologies are updated every other day, however the architecture must be updated constantly to keep up with the latest technologies.

The overall success was in designing and developing a real-world project collaborating effectively within teamwork. Moving forward, the goal is to continually update the demo version to evolve into a fully-fledged production project and the deadline for a prod version is at the end of April 2024. Subsequently, we plan to develop a user interface (UI) for configuring campaigns and providing a comprehensive campaign report.

References

Cheah, D. 2019. How to work with Servlet, JSP and JDBC? Available at: https://medium.com/@tattwei46/how-to-work-with-servlet-jsp-and-jdbc-fcc568a6a57b . Accessed: 10-02-2024.

Hibernate ORM. Online document. Available at: https://hibernate.org/orm/ . Accessed: 07-02-2024.

Jaiswal Sonoo. Available at: https://www.javatpoint.com/hibernate-tutorial Accessed: 07-02-2024.

Jaiswal Sonoo. Online document. Available at: https://www.javatpoint.com/java-tutorial. Accessed: 05-02-2024.

JUnit FAQ. Online document. Available at: https://junit.org/junit4/faq.html. Accessed: 10-03-2024.

JGroups documentation. Online document. Available at: http://www.jgroups.org/ Accessed: 20-02-2024.

Tomcat documentation. Online document. Available at: https://tomcat.apache.org/tomcat-9.0-doc/index.html Accessed: 29-02-2024.

What is postgresql. Online decument. Available at: https://www.postgresql.org/. Accessed: 06-02-2024

Willbold Manuela. Online document. Available at: https://www.clickdo.co.uk/internet-marketing/marketing-techniques-for-igaming-industry/ Accessed: 10-01-2024.

Appendices

Appendix 1: Source code

In the source code below, we are having the first hardcoded version of PromoDao, PromoServlet, PromoEngine and PromoCampaign classes.

```
/**
* Entity class representing a free spin campaign. This class maps to
the
 * "promocampaign" table in the database.
 */
@TypeDef(name = "MetadataJsonbType", typeClass =
MetadataJsonbUserType.class)
@TypeDef(name = "PrizesJsonbType", typeClass =
PrizesJsonbUserType.class)
@Entity
@Table(name = "promocampaign")
public class PromoCampaign
{
        // Represents the unique identifier for the campaign.
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id")
        public Integer id;

        // Represents games that configured by campaign.
        @Column(name = "games", nullable = false)
        public String games;

        // Represents the start time of the campaign.
        @Column(name = "starttime", nullable = false)
        @JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ssX")
        public Timestamp startTime;

        // Represents the end time of the campaign.
        @Column(name = "endtime", nullable = false)
        @JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ssX")
        public Timestamp endTime;

        // Represents the prizes associated with the campaign in JSON
        format (jsonb).
        @Column(name = "prizes", columnDefinition = "jsonb", nullable =
        false)
        @Type(type =
        "com.airdice.gamesystem.promoserver.json.PrizesJsonbUserType")
        public Prizes prizes;

        // Represents the minimum bet level allowed for the c campaign.
        @Column(name = "minbet", precision = 19, scale = 2)
        public BigDecimal minBet;

        // Represents the maximum bet level allowed for the campaign.
```

```
@Column(name = "maxbet", precision = 19, scale = 2)
public BigDecimal maxBet;

// Represents the validity date of the campaign.
@Column(name = "validitydate")
@JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ssX")
public Timestamp validityDate;

// Represents the bet level for free spins in the campaign.
@Column(name = "fsbetlevel", precision = 19, scale = 2)
 public BigDecimal fsBetLevel;

 // Represents the customer, which have the campaign.
 @Column(name = "customer", nullable = false, length = 255)
 public String customer;

 /**
   * Represents additional metadata associated with the campaign
       in JSON
  * format (jsonb) like campaign name and rules link
  *
  */
@Column(name = "metadata", columnDefinition = "jsonb", nullable
= false)
@Type(type =
"com.airdice.gamesystem.promoserver.json.MetadataJsonbUserType"
)
public Metadata metadata;

 /**
  * Get the ID of the campaign.
  *
  * @return The campaign ID.
  */
 public Integer getCampaignId()
 {
    return id;
 }

 /**
  * Set the ID of the campaign.
  *
  * @param campaignId The campaign ID to set.
  */
  public void setCampaignId(Integer campaignId)
  {
    this.id = campaignId;
  }

 /**
  * Get the games associated with the campaign.
  *
  * @return The games.
  */
  public String getGames()
 {
   return games;
 }

 /**
  * Set the games associated with the campaign.
 *
 * @param games The games to set.
 */
public void setGames(String games)
 {
   this.games = games;
 }

/**
 * Get the start time of the campaign.
 *
 * @return The start time.
 */
 public Timestamp getStartTime()
 {
   return startTime;
 }

/**
 * Set the start time of the campaign.
 *
 * @param startTime The start time to set.
 */
 public void setStartTime(Timestamp startTime)
 {
   this.startTime = startTime;
 }


/**
 * Get the end time of the campaign.
 *
 * @return The end time.
 */
 public Timestamp getEndTime()
 {
   return endTime;
 }

/**
 * Set the end time of the campaign.
 *
 * @param endTime The end time to set.
 */
 public void setEndTime(Timestamp endTime)
 {
   this.endTime = endTime;
 }

/**
 * Get the prizes associated with the campaign.
 *
 * @return The prizes.
 */
 public Prizes getPrizes()
 {
   return prizes;
 }

/**
 * Set the prizes associated with the campaign.
 *
 * @param prizes The prizes to set.
 */
 public void setPrizes(Prizes prizes)
 {
   this.prizes = prizes;
 }

/**
 * Get the minimum bet amount for the campaign.
 *
 * @return The minimum bet amount.
 */
 public BigDecimal getMinBet()
 {
   return minBet;
 }

/**
 * Set the minimum bet amount for the campaign.
 *
 * @param minBet The minimum bet amount to set.
 */
 public void setMinBet(BigDecimal minBet)
 {
   this.minBet = minBet;
 }

/**
 * Get the maximum bet amount for the campaign.
 *
 * @return The maximum bet amount.
 */
 public BigDecimal getMaxBet()
 {
   return maxBet;
 }

/**
 * Set the maximum bet amount for the campaign.
 *
 * @param maxBet The maximum bet amount to set.
 */
 public void setMaxBet(BigDecimal maxBet)
 {
   this.maxBet = maxBet;
 }

/**
 * Get the validity date of the campaign.
 *
 * @return The validity date.
 */
 public Timestamp getValidityDate()
 {
   return validityDate;
 }

/**
 * Set the validity date of the campaign.
 *
 * @param validityDate The validity date to set.
 */
 public void setValidityDate(Timestamp validityDate)
 {
    this.validityDate = validityDate;
 }

/**
 * Get the bet level associated with the campaign.
 *
 * @return The bet level.
 */
 public BigDecimal getFsBetLevel()
 {
    return fsBetLevel;
 }

/**
 * Set the bet level associated with the campaign.
 *
 * @param fsBetLevel The bet level to set.
 */
 public void setFsBetLevel(BigDecimal fsBetLevel)
 {
    this.fsBetLevel = fsBetLevel;
 }

/**
 * Get the customer associated with the campaign.
 *
 * @return The customer.
 */
 public String getCustomer()
 {
    return customer;
 }

/**
 * Set the customer associated with the campaign.
 *
 * @param customer The customer to set.
 */
 public void setCustomer(String customer)
 {
    this.customer = customer;
 }

/**
 * Get the metadata associated with the campaign.
 *
 * @return The metadata.
 */
 public Metadata getMetadata()
 {
    return metadata;
 }

/**
 * Set the metadata associated with the campaign.
 *
 * @param metadata the metadata to set.
```

```
     */
     public void setMetadata(Metadata metadata)
     {
        this.metadata = metadata;
     }
}


/**
 * The PromoDAO class provides data access functionality for
handling promo
 * campaigns and associated data in the application. It utilizes
Hibernate for
 * database interaction.
 */
public class PromoDao
{
    private DbTransactionManager transactionManager;
    private SessionFactory sessionFactory;
    List<PromoCampaign> promoList = new ArrayList<>();

    public PromoDao(){}

        public PromoDao(DbTransactionManager transactionManager)
        {
           this.transactionManager = transactionManager;
        }

        public SessionFactory buildSessionFactory()
        {
            Properties dbprops = new Properties();

            // Load properties file from the classpath
           try (InputStream is = getClass().getClassLoader()
           .getResourceAsStream("hibernate.propertis"))
          {
            if (is == null)
            {
                  throw new IOException("Cannot find
                  'hibernate.properties' file in classpath");
            }
            dbprops.load(is);

            Configuration config = new Configuration();
            config.setProperties(dbprops);
            config.addAnnotatedClass(PromoCampaign.class);

           StandardServiceRegistryBuilder srb = new
           StandardServiceRegistryBuilder().applySettings(config.g
           etProperties());
           sessionFactory =
           config.buildSessionFactory(srb.build());

           System.out.println("connected to db... ");

            return sessionFactory;
        }
        catch (IOException e)
        {
             throw new RuntimeException("Error while initializing
                   PromoDAO", e);
        }
    }

    /**
     * store a configured freespins campaign to the database.
     *
     * @param campaign The FSCampaign object to be saved.
       * @throws ParseException If an error occurs while parsing
       date-related
     *             data.
     */
       public Integer savePromos(PromoCampaign campaign) throws
       ParseException
    {
        transactionManager.getCurrentSession().save(campaign);

        // Flush to reveal constraint violations
        transactionManager.getCurrentSession().flush();

        return campaign.getCampaignId();
    }

    /**
       * Retrieves a list of freespins giveaway campaigns from the
       database.
     *
         * @return List of FSCampaign objects representing
    promotional campaigns.
     * @throws JsonProcessingException
     */
    public List<PromoCampaign> getPromos() throws
    JsonProcessingException
    {
            String hql = "FROM PromoCampaign";

           Query<PromoCampaign> query =
           transactionManager.getCurrentSession().createQuery(hql,
           PromoCampaign.class);

            return query.list();
    }
}

@Override
protected void doPost(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException
    {
        try
        {
            String method = "";
            String response;
            int responseCode = HttpServletResponse.SC_OK;
            try
            {
                method = getMethod(req.getRequestURI());

                transactionManager.startTransaction();
                try
                {
                    PromoCampaign promoCampaign;
                    switch (method)
                {
                    case "CreatePromo":
                      StringBuilder requestBodyBuilder = new
                      StringBuilder();
                      BufferedReader reader = req.getReader();
                      String line;
                       while ((line = reader.readLine()) !=
                              null)
                      {
                          requestBodyBuilder.append(line);
                      }
                      String requestBody =
                      requestBodyBuilder.toString();


                        promoCampaign =
                        jsonMapper.readValue(requestBody,
                        PromoCampaign.class);
                        promoDao.savePromos(promoCampaign);
                        break;
                    default:
                      throw new
                      IllegalArgumentException("Unknown
                      method" + method);
                }

               response =
               jsonMapper.writeValueAsString(promoCampaign);
               transactionManager.commit();
            }
            finally
            {
                transactionManager.endTransaction();
            }
        }
        catch (Exception e)
        {
            ErrorCode errorCode = ErrorCode.UNKNOWN_ERROR;

              res.sendError(HttpServletResponse.SC_INTERNAL_SER
              VER_ERROR, "Error while saving campaign");
            e.printStackTrace();
              response =
              jsonMapper.writeValueAsString(ResponseBase.ofErro
              r(errorCode));
        }

        // Write response
        res.setContentType("application/json");
        res.setCharacterEncoding("utf-8");
        res.setStatus(responseCode);
        res.getWriter().write(response);
    }
    finally
    {
    }
}


/**
```

```
* @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse
         *      response)
         */
        @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
    res) throws ServletException, IOException
            {
            try
            {
                String method = "";
                String response = null;
                int responseCode = HttpServletResponse.SC_OK;
                List<PromoCampaign> promos =  new ArrayList<>();
                PromoWinsResponse promoWins = new PromoWinsResponse();
                try
                {
                    method = getMethod(req.getRequestURI());
                    transactionManager.startTransaction();
                    promoEngine.getPromoCampaign();
                    try
                    {
                        // Initialize or retrieve GameRoundData here
                        HttpSession session = req.getSession(true);
                       GameRoundData grd = (GameRoundData)
                      session.getAttribute("gameRoundData");

                        if (grd == null) {

                               grd = new GameRoundData("customer_test",
                                       "RandomReels", "adaham");

                               session.setAttribute("gameRoundData",
                               grd);
                        }

                        switch (method)
                        {
                            case "getPromos":
                               Timestamp currentTime =
                               Timestamp.from(Instant.now());
                               promos = promoEngine
                               .getActivePromos("customer_test",
                               "RandomReels",
                               currentTime, new BigDecimal(20.00));
                               response =
                               jsonMapper.writeValueAsString(promos);
                              break;
                           case "pollPromoWins":
                               Timestamp time =
                               Timestamp.from(Instant.now());
                               promoWins =
                               promoEngine.awardPromoWins(grd, new
                               BigDecimal(20.00), time);
                               response =
                               jsonMapper.writeValueAsString(promoWins)
                               ;
                              break;
                           default:
```

```
                          throw new
                          IllegalArgumentException("Unknown method
                          " + method);
                    }

                    transactionManager.commit();
                }
                finally
                {
                    transactionManager.endTransaction();
                }
            }
            catch (Exception e)
            {
                ErrorCode errorCode = ErrorCode.UNKNOWN_ERROR;

                  res.sendError(HttpServletResponse.SC_INTERNAL_SER
                  VER_ERROR,"Error while saving campaign");
                   e.printStackTrace();
                  response =
                  jsonMapper.writeValueAsString(ResponseBase.ofErro
                  r(errorCode));
            }

            // Write response
            res.setContentType("application/json");
            res.setCharacterEncoding("utf-8");
            res.setStatus(responseCode);
            res.getWriter().write(response);
        }
        finally
        {
        }
    }


    /**
     * Returns currently active promos for the customer and game.
Active means
     * startTime <= time < endTime.
     *
     * @throws JsonProcessingException
     */
public List<PromoCampaign> getActivePromos(String customerId,
String gameId, Timestamp time) throws JsonProcessingException
 {

     List<PromoCampaign> activePromos = new ArrayList<>();

     if (promos != null && !promos.isEmpty())
     {
        for (PromoCampaign promo : promos)
        {

           if (promo.getGames() != null &&
           promo.getGames().contains(gameId)
           && time.compareTo(promo.getStartTime()) >= 0 &&
           time.compareTo(promo.getEndTime()) < 0)
           {
              activePromos.add(promo);
                }
            }
        }
        return activePromos;
    }


    /**
* Returns currently active campaigns for the customer, game and
bet level.
     * Active means startTime <= time < endTime.
     *
     * @param normBetLevel.
     * @throws JsonProcessingException
     */
public List<PromoCampaign> getActivePromos(String customerId,
String gameId, Timestamp time, BigDecimal normBetLevel) throws
JsonProcessingException
{
    List<PromoCampaign> activePromos = getActivePromos(customerId,
    gameId, time);

    Iterator<PromoCampaign> iter = activePromos.iterator();
    PromoCampaign promo;
    while (iter.hasNext())
    {
           promo = iter.next();
           if (promo.getMinBet().compareTo(normBetLevel) > 0 ||
           promo.getMaxBet().compareTo(normBetLevel) < 0)
                   iter.remove();
    }

    return activePromos;
}


// Award promos every second game round
public PromoWinsResponse awardPromoWins(GameRoundData grd,
BigDecimal betLevel, Timestamp time) throws
JsonProcessingException
{
    PromoWinsResponse promoWins = new PromoWinsResponse();

     // Retrieve active promos for the customer, game, and current
    time
    List<PromoCampaign> activePromos =
    getActivePromos(grd.getCustomer(), grd.getGameId(), time,
    betLevel);

    if (grd.isNextWin)
        {
           try
            {
                for (PromoCampaign promo : activePromos)
                {
                        promoWins.setBetLevel(betLevel);

                           promoWins.setFreespinsAmount(promo.getPr
                           izes().getPrizePerDrop().get(0));

                   // Toggle isNextWin to false for the next round
                    grd.setNextWin(false);
        }

    }
    catch (Exception e)
    {
        System.out.println(
                          "Error during awardCampaignWins:
          customer {}, game {}"             +
          grd.getCustomer() + grd.getGameId());
    }
} else {
    grd.setNextWin(true);
    return null;
}

return promoWins;
}
```

Listing 5. Promo server demo source code.