ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი
კომპიუტერულ მეცნიერებათა დეპარტამენტი

იოსები გობრონიძე

პრომო-კამპანიების მართვის სისტემა - Promo Campaigns Management System

სამაგისტრო პროგრამა: ინფორმაციული სისტემები

სამაგისტრო ნაშრომი შესრულებულია ინფორმაციულ სისტემებში
მეცნიერების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელები:
გია სირბილაძე,
ფიზ.-მათ. მეცნიერებათა დოქტორი,
სრული პროფესორი;
ირინა ხუციშვილი,
ფიზ.-მათ. მეცნიერებათა კანდიდატი,
ასოცირებული პროფესორი.

თბილისი

2019

ანოტაცია

სამაგისტრო ნაშრომი წარმოადგებს პრომო-კამპანიების (უფრო ზოგადად, წამახალისებელი გათამაშებების) მართვის სისტემის აღწერას. სისტემის შექმნის აუცილებლობა გამოწვეულია მრავალი პრაქტიკული პრობლემის გადასაჭრელად. იგი დღეს-დღეობითაც ემსახურება რეალურ სამყაროში კლიენტების მოთხოვნებს და ასრულებს მნიშვნელოვან როლს ზოგიერთი კომპანიის განვითარებაში. სისტემის ფუნქციონირების მთავარი ასპექტები პრაქტიკული გამოცდილებაზეა დაფუძნებული და ითვალისწინებს ახალი გამოცდილების გამოყენებას. ეს კი უზრუნველყოფს კლიენტზე მორგებულობას და დამკვეთის მაქსიმალურ კომფორტს.

სისტემის ძირითადი შესაძლებლობები შეგვიძლია შემდეგ ნაწილებად დავყოთ:

- კონკრეტული კომპანიებისთვის არხებისა და მომხმარებლების შექმნა, რომლებიც შემდეგ უზრუნველყოფენ აპლიკაციის გამართულ მუშაობას;
- პრომო-კამპანიისა და შესაბამისი ვაუჩერების შექმნა/გენერაცია;
- არსებული კამპანიებისა და ვაუჩერების შესახებ ინფორმაციის ინტეგრირებული არხისთვის ვებ-სერვისით მიწოდება;
- სტატისტიკური მონაცემების დათვალიერება ვებ-აპლიკაციით ან ინფორმაციის რეპორტებში ასახვა;

რადგანაც სისტემა პირდაპირ კავშირშია მომხმარებლებთან და ასევე სისტემას პირდაპირი კავშირი აქვს მატერიალურ ღირებულებებთან (მოგებული პრომო-კოდი), ძალიან დიდი მნიშვნელობა ენიჭება ისეთ ფაქტორებს, როგორიცაა სისტემის სანდოობა, ხელმისაწვდომობა, საიმედოობა, უსაფრთხოება, დაცულობა და სხვა. სწორედ ამ ფაქტორების მაღალი ხარისხით წარმოჩენაა სისტემის მთავარი მიღწევა.

Annotation

The Master's Thesis illustrates a system of promo campaigns (to be more general – promotional draws and competitions) management. The system development is necessary for solving some real problems. Even today, it serves the clients' needs and plays its role in the development of several companies. The main functioning aspects of the system are based on practical experience and from time to time some improvements are done based on new experiences, which ensure the maximal comfort of the order-giving customer and fit best the client needs.

The basic tasks of the system can be divided in the following parts:

- Create channels and users for specific companies which ensure the proper work of the application.
- Create/generate the promo campaigns and corresponding vouchers.
- Deliver the information about the existing campaigns and vouchers through the web-service to the integrated channel.
- View the statistical data in the web-application or show the information in the reports.

Regarding the fact that the system is directly connected with the users and tangible assets (won promo code), the following factors have great importance - system reliability, safety, security and etc. Exactly the high assessment of these factors is the main achievement of the system.

სარჩევი

1 მონაცემთა ბაზა

1.1 მონაცემთა ბაზის ტიპი, კავშირის შუალედური დამხმარე ბიბლიოთეკები

ინფორმაციული სისტემის აპლიკაცია რა თქმა უნდა მთლიანად ეფუძნება ინფორმაციას, რომელიც გროვდება სისტემის ადმინისტრატორებისა და მომხმარებლების მიერ. სწორედ ამ ინფორმაციის გაცვლა-გამოცვლა ქმნის აპლიკაციას, რომელიც ემსახურება კლიენტებს. შესაბამისად ინფორმაციის შენახვა და დამუშავება არის ძალიან მნიშვენლოვანი. ასევ მნიშვნელოვანია, პროექტის დაწყების ეტაპზევე სწორად გაითვალოს თუ რა სახით და სად უნდა შეინახოს აღნიშნული მონაცემები.

განხილული პროგრამულის უზრუნველყოფა - პრომო-კამპანიების მართვის სისტემა მონაცემების შესანახად იყენებს რეალაციურ მონაცემთა ბაზას, კერძოდ კი Microsoft SQL Server-ს. რელაციური ბაზის არჩევა განაპირობა სისტემის თავისებურებებმა. მონაცემები, რომელიც მიმოიცვლება არის რეალაციური ტიპის, ყველა ობიექტი მჭიდროდ არის ერთმანეთთან დაკავშირებული, შესაბამისად რელაციური ტიპის ბაზის არჩევა გარდაუვალი იყო.

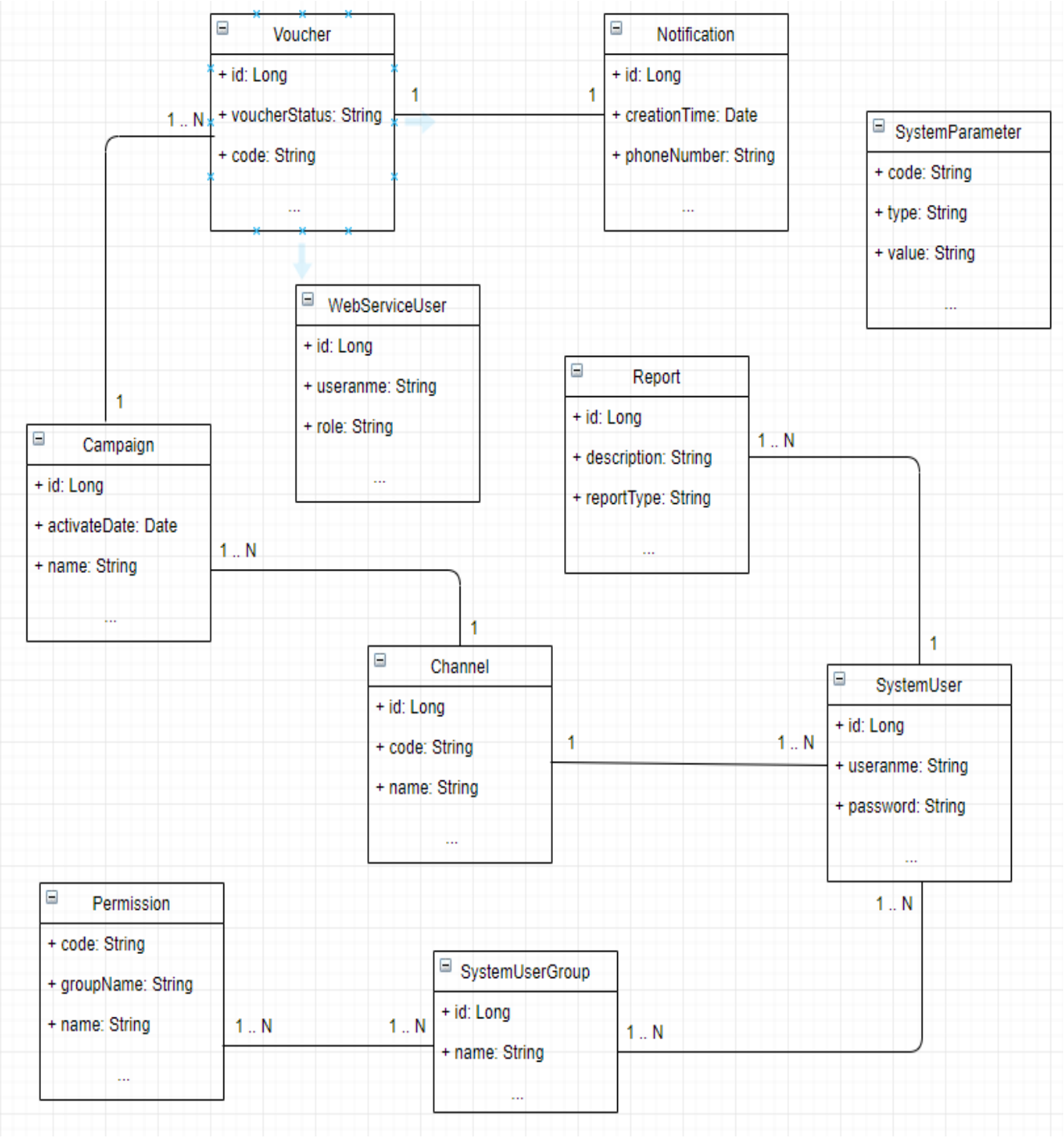
აპლიკაციასა და მონაცემთა ბაზას შორის კავშირი მყარდება ORM Tool-ს დახმარებით. ეს დამხმარე საშუალება არის JPA(Java Persistence API), რომელიც ამ კონკრეტულ შემთხვევაში უზრუნველყოს Java-ს და Microsoft SQL Server-ს შორის ინფორმაციის მიმოცვლას. შესაბამისად აღარ იქმნება პირდაპირი სკრიპტების წერის საშუალება, გამოყენებული ლიტერატურა - Mike Keith and Merrick Schnicariol, Pro JPA 2, Mastering the Java Persistence API, Apress 2016;

ბაზასთან ურთიერთობისას რა თქმა უნდა გათვალისწინებულია ძირეული საკითხები, რომელიც უზრუნველყოფს პირველ რიგში გარანტირებულ სწორად მუშაობას და სწრაფქმედებას. ნებისმიერი ცვლილება არის ტრანზაქციული, რაც უზრუნველყოფს მონაცემების მთლიანობას, არ ტოვებს საშიშროებას, რომ გარკვეული მონაცემი დაკომიტდეს მასზე დამოკიდებული სხვა მონაცემის ცვლილების გარეშე.

ასევე ბაზის ცხრილების გარკვეულ ველებზე ადევს კლასტერული ინდექსები. ინდექსების დადების აუცილებლობა გამოწვეულია დატვირთული სელექთების არსებობით. ეს საკითხი ყველაზე მნიშვნელოვნად ვლინდება რეპორტების ამოღების დროს. სისტემას შეუძლია გაუმკლავდეს ძალიან დიდი რაოდენობით მონაცემებს, ამიტომაც ინდექსების გარეშე პრაქტიკულად შეუძლებელი იქნებოდა რეპორტების ამოღება, თუ მომხმარებლის ინტერფეისიდან ინფორმაციის გაფილტვრა.

1.2 UML დიაგრამა

ქვემოთ მოცემულ დიარგამაში ასახულია სისტემაში გამოყენებული ბაზის ცხრილები და მათ შორის ურთიერთკავშირი. გარკვეულ ცხრილებში ველების სიმრავლის გამო აქ ასახულია მხოლოდ სამ-სამი კომპნენტი, გამოყენებული ლიტერატურა - Ramez Elmasi, Shamkart B. Navathe, Fundamentals of database systems seventh edition.



განვიხილოთ რამდენიმე საყურადღებო და ძირეული მომენტი, რომელიც ასახულია მოცემულ დიაგრამაზე:

- ხაზგასასმელია, რომ რეპორტებისთვის გამოყენებულია მხოლოდ ერთი ცხრილი და ყველა განსხვავებული ტიპის რეპორტი მასშია მოთავსებული, განსხვავება იმართება პარამეტრით **reportType**. ასევე გასათვალისწინებელია, რომ **Report**-ს და **SystemUser**-ს შორის არის კავშირი, რაც იძლება საშუალებას, კონკრეტული რეპორტი ეკუთვნოდეს მხოლოდ ერთ მომხმარებელს და სხვას არ შეეძლოს მისი ნახვა და შეცვლა.

- **Permission**, **SystemUserGroup** და **SystemUser** ცხრილების მოცემული ურთიერთკავშირი მთლიანად იძლევა საშუალებას, სისტემა იმართებოდეს განსხვავებული როლების მქონე მომხმარებლებისგან, რომელთათვისაც უფლებების მატრიცის გაწერა შესაძლებელია პრივილეგირებული მომხმარებლის მიერ.

- მთავარი კავშირი გამოისახება **Campaign** და **Voucher** ცხრილებს შორის. სწორედ ეს კავშირი განაგებს სისტემის ძირეულ ფუნქციონალს - კამპანიისთვის ვაუჩერის მოთხოვნისას მოიძიოს შესაბამისი ვაუჩერი და გაამზადოს მომხმარებლისთვის გადასაცემად.

- **WebServiceUser** და **SystemParameter** ცხრილები არის ყველასგან დამოუკიდებელი. მათი მოვალეობაა შეინახოს ინფორმაცია რომელიც მნიშვნელოვანია სისტემის კონფიგურაციისთვის, სწორედ ამ ცხრილების დახმარებით ხდება აპლიკაციის კონფიგურირება, ისე რომ მორგებული იყოს ნებისმიერი კლიენტისთვის.

- **Voucher** და **Notification** ცხრილს შორის არის ერთი ერთთან კავშირი, რაც გვანიშნებს, რომ შესაძლებელი იყოს ამ ცხრილების გაერთიანება. გაყოფის გადაწყვეტილება გამოწვეულია კვლავ ზოგადობის პრინციპის გამო. თავის თავში **Notification** ცხრილი აერთიანებს როგორც ვაუჩერებისთვის განკუთვნილი შეტყობინების გაგზავნის ინფორმაციას, ასევე შესაძლებელია მისი გამოყენება სხვა ტიპის შეტყობინებებისთვის.

2 PCMS პროცესინგი

როგორც ყველა აპლიკაცია, ასევე PCMS-ის მთავარი სირთულე და მნიშვნელოვანი გამოწვევები დევს პროცესინგის ნაწილში. ეს არის ის ნაწილი სადაც განთავსებულია კამპანიების შექმნის, ვაუჩერების დამატების, მათი გაცემის და საჭიროების შემთხვევაში გაუქმების ურთიერთქმედებები. სირთულე სწორედ ამ ურთიერთქმედებაში მდგომარეობს.

პროცესინგის მაქსიმალური აჩქარებისთვის გამოყენებულია ქეშირება. სისტემაში აპლიკაციის მეხსიერებაში ინახება ნებისმიერი ინფორმაცია, რომელიც მონაწილეობს ვაუჩერების გაცემაში; ესენია კამპანიები, ვაუჩერები და არხები. ქეშირება აწყობილია ისე, რომ ყოველთვის სინქრონულია ბაზაში არსებულ ინფორმაციასთან და ეს სინქრონიზაცია თავიდან ეშვება მომხმარებლის ინტერფეისიდან ნებისმიერი ცვლილების შემდეგ. ასეთი მიდგომის წყალობით, მომხმარებლისგან მოთხოვნილ არცერთ ქმედებაზე არ გვიწევს ბაზაში ინფორმაციის გადამოწმება ყველაფერი სრულდება მეხსიერებიდან, რაც რა თქმა უნდა საგრძნობლად ზრდის აპლიკაციის სწრაფმოქმედებას და უზრუნველყოფს ნებისმიერი ქმედების მყისიერ შესრულებას.

```
@TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
public void campaignUpdated(@Observes(during = TransactionPhase.AFTER_SUCCESS) Campaign campaign) {
    String lockKey = CAMPAIGN_CHANGE;
    ReentrantLock lock = null;
    try {
        lock = actionLocksManager.lockInvocation(lockKey);
        for (List<Campaign> campaignsList : channelCampaignsMap.values()) {
            campaignsList.removeIf(c -> c.getId() == campaign.getId());
        }
        if (campaign.getStatus() != CampaignStatus.ACTIVE && campaign.getStatus() != CampaignStatus.ADDING_VOUCHERS
                && campaign.getStatus() != CampaignStatus.STOPPED_VOUCHER_ADDITION) {
            campaignReservableVouchersMap.remove(campaign.getId());
            activeCampaigns.remove(campaign.getId());
            campaignIssuedPhoneNumbersMap.remove(campaign.getId());
        } else {
            boolean isNewCampaign = !activeCampaigns.containsKey(campaign.getId());
            initActiveCampaign(campaign);
            if (isNewCampaign) {
                initActiveVouchers(campaign);
            }
        }
    } finally {
        actionLocksManager.unlockInvocation(lock, lockKey);
    }
}
```

პროცესის ასაჩქარებლად ასევე პროცესინგში რამდენიმე ადგილას გამოყენებულია ასინქრონული მექანიზმები. მაგალითად მთლიანად ასინქრონულია შეტყობინებების გაგზავნა. ვაუჩერის გაცემისას კლიენტი არ ელოდება შეტყობინების გაგზავნის შედეგს, ის წარმატებულ პასუხს იღებს და სისტემა შემდეგ ცდილობს შეტყობინების გაგზავნას. რა თქმა

უნდა აქ ჩნდება საშიშროება, რომ კლიენტმა პროცესი წარმატებულად ჩათვალოს, მაგრამ ვერ მიიღოს სასურველი შეტყობინება. ამის გადასაჭრელად სისტემაში არსებობს შეტყობინების ხელმეორედ გაგზავნის შესაძლებლობა, რომელიც გამოიყენება ასეთ პრობლემურ სიტუაციებში. შეტყობინებების გაგზავნა თავის მხრივ არის სრულიად ცალკე პროცესი, ამისთვის სისტემა დაკავშირებულია Messaging System-თან, რომელსაც აწვდის სასურველ შეტყობინებებს და ის უზრუნველყოფს პროცესის დანარჩენ ნაწილს.

```
@Override
public boolean saveNotifications(List<String> phoneNumbers, Map<String, String> params) {
    if (StringUtils.isEmpty(params.get("lastTerm"))) {
        params.put("lastTerm", termlessTextForSMS.value());
    }
    List<Notification> notifications = getNotifications(phoneNumbers, params);
    logger.info("Start saving notifications size[" + notifications.size() + "]");
    boolean result = false;
    try {
        tx.begin();
        if (!notifications.isEmpty()) {
            for (Notification notification : notifications) {
                em.persist(notification);
            }
        }
        tx.commit();
        result = true;
    } catch (Exception ex) {
        logger.error("Error occurred during create notifications", ex);
        rollback();
    }
    logger.info("Saved notifications");
    return result;
}
```

რა თქმა უნდა სასურველია, სისტემა არასდროს გამოვიდეს წყობიდან, თუმცა ყოველთვის არსებობს ავარიული გამორთვის საშიშროება. შესაბამისად პროცესინგი ისე უნდა იყოს აწყობილი, რომ მინიმალური იყოს ავარიული გამორთვის მიერ მიყენებული ზიანი. სწორედ ამისთვის პროცესინგში დამატებულია სერვერის გამორთვისას ქეშირებაში არსებული დროებითი ინფორმაციის ფაილში გატანა. ასეთი ინფორმაცია არის დარეზერვებული ვაუჩერები, რადგან ის არ ინახება ფაილში (არის მხოლოდ დროებით მეხსიერებაში) და სისტემის გაჩერებისას ავტომატურად გადადის დროებით ფაილში. აპლიკაციის თავიდან ჩართვისას ეს ინფორმაცია იმავე ფაილიდან აისახება მიმდინარე დროებით მეხსიებერაში

```java
private void dumpReservedVouchers() {
    if (savedReservedVouchersFilePath.value() != null) {
        Map<Long, ReservedVoucher> reservedVouchersMapCopy = new HashMap<>(reservedVouchersMap);
        log.info("Dumping Reserved vouchers to file. count=" + reservedVouchersMapCopy.size()
                + " file path=" + savedReservedVouchersFilePath.value());
        try (BufferedWriter bf = new BufferedWriter(new FileWriter(savedReservedVouchersFilePath.value()))) {
            int i = 0;
            for (ReservedVoucher reservedVoucher : reservedVouchersMapCopy.values()) {
                bf.write(String.valueOf(reservedVoucher.voucher.getId()));
                if (i != reservedVouchersMapCopy.size() - 1) {
                    bf.write(DELIMITER);
                }
                i++;
            }
        } catch (IOException ex) {
            log.error("Error while writing reserved vouchers to file", ex);
        }
    } else {
        log.warn("No file path found for Dumping Reserved vouchers");
    }
}
```

სისტემის მთავარ ობიექტებზე, კამპანიაზე და ვაუჩერზე პროცესინგი მიმდინარეობს რამდენიმე მეთოდით:

- List<Campaign> getActivePromoCampaigns(String channelCode) {}
- HashMap<Long, Long> reserverPromoCode(String channelCode, Set<Long> campaignIds) {}
- void issuePromoCodeManual(String channelCode, long voucherId, String smsText, String contactNumber, String username) {}
- String issuePromoCode(String channelCode, long campaignId, long promoCodeId, String smsText, String contactNumber, Sring stationId, String serviceName) {}
- void cancelVoucher(long voucherId, String username) {}
- void campaignUpdated(Campaing campaign) {}

მაქსიმალური სისწრაფისთვის საჭიროა ეს მეთოდები იყოს პარალელური, თუმცა აქ იქმნება თანაკვეთის პრობლემა. არსებობს საშიშროება, რომ ერთ ობიექტზე ერთდროულად იმოქმედოს რამდენიმე მეთოდმა, რაც სავარაუდოდ გამოიწვევს პრობლემებს. ამის გადასაჭრელად გამოყენებულია ე.წ **ReentrantLock**, რომლის დახმარებითაც იბლოკება მხოლოდ კონკრეტული ობიექტები. ამით მიღწეულია მაქსიმალური სისწრაფეც და გადაჭრილია ერთ ობიექტზე რამდენიმე ნაკადით მოქმედების პრობლემაც.

3 PCMS როგორც ინფორმაციის მიმწოდებელი

PCMS მთავარი დანიშნულება არის კამპანიების მენეჯმენტი, რაც თავისთავად მოიცავს ინფორმაციის არხებისთვის მიწოდებას. ინფორმაციის გაცვლისთვის გამოყენებულია SOAP ვებ სერვისები:

3.1 getPromoCampaigns

მეთოდი აბრუნებს სისტემაში არსებულ აქტიურ და გასააქტიურებელ კამპანიების სიას, გადაცემული არხის კოდის საფუძველზე (მხოლოდ იმათ, რომელიც გაწეირილია შესაბამისი არხისთვის).

გადასაცემი პარამეტრები

პარამეტრი	ტიპი	აღწერა
channelCode	String	PCMS სისტემაში არსებული შესაბამისი გაყიდვის არხის კოდი

დასაბრუნებელი მნიშვნელობები

პარამეტრი	ტიპი	აღწერა
id	Number	უნიკალური იდენტიფიკატორი
name	String	დასახელება
startDate	Date	დაწყების თარიღი
endDate	Date	დასრულების თარიღი
prioriy	Enum	პრიორიტეტი. შესაძლო მნიშვნელობები[**LOW**, **MEDIUM**, **HIGH**]
voucherExpirationDate	Date	ვაუჩერის ვადის გასვლის თარიღი

3.2 reservePromoCode

მეთოდი არეზერვებს პრომო კოდებს (ვაუჩერი) და აბრუნებს მათ იდენთიფიკატორებს. თუ კონკრეტული პარამეტრისთვის ვერ მოხერხდა პრომო კოდის დარეზერვება, შესაბამისი მნიშვნელობა არ ბრუნდება.

გადასაცემი პარამეტრები

პარამეტრი	ტიპი	აღწერა
channelCode	String	PCMS სისტემაში არსებული შესაბამისი გაყიდვის არხის კოდი
campaignIds	List<Number>	კამპანიების იდენთიფიკატორების სია,

		რომლისთვისაც რეზერვდება პრომო კოდები

დასაბრუნებელი მნიშვნელობები

პარამეტრი	ტიპი	აღწერა
campaignId	Number	კამპანიის უნიკალური იდენტიფიკატორი
prmoCodeId	String	პრომო კოდის(ვაუჩერის) უნიკალური იდენტიფიკატორი

3.3 issuePromoCode

მეთოდი გაცემს პრომო კოდს (ვაუჩერს) უნიკალური იდენთიფიკატორზე დაყრდნობით და აბრუნებს მისთვის განკუთვნილ კოდს. აუცილებელია, რომ შესაბამისი ვაუჩერი იყოს დარეზერვებული გარკვეული(სისტემური პარამეტრით გაწერილი) დროის მონაკვეთში.

გადასაცემი პარამეტრები

პარამეტრი	ტიპი	აღწერა
channelCode	String	PCMS სისტემაში არსებული შესაბამისი გაყიდვის არხის კოდი
campaignId	Number	კამპანიის უნიკალური იდენტიფიკატორი
promoCodeId	Number	პრომო კოდის(ვაუჩერის) უნიკალური იდენტიფიკატორი
smsText	String	გასაგზავნი შეტყობინების ტექსტი
contactNumber	String	საკონტაქტო ნომერი, სადაც უნდა გაიგზავნოს შეტყობინება
stationId	Number	ადგილის(წერტილის) იდენთიფიკატორი, საიდანაც გაიცა პრომო კოდი
serviceName	String	სერვისის სახელი, რისთვისაც გაიცა პრომო კოდი

დასაბრუნებელი მნიშვნელობები

პარამეტრი	ტიპი	აღწერა
code	String	პრომო კოდის(ვაუჩერის) კოდი

3.4 სერვისის გამოძახების მაგალითი

განვიხილოთ სერვისის გამოძახება getPromoCampaigns მეთოდის მაგალითზე. სერვისის რექვესთს ექნება შემდეგი სახე:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:pcms="http://pcms.azry.com">
    <soapenv:Header/>
    <soapenv:Body>
        <pcms:getPromoCampaigns>
            <ChannelCode>amazon</ChannelCode>
        </pcms:getPromoCampaigns>
    </soapenv:Body>
</soapenv:Envelope>
```

შესაბამისი რესფონსი:

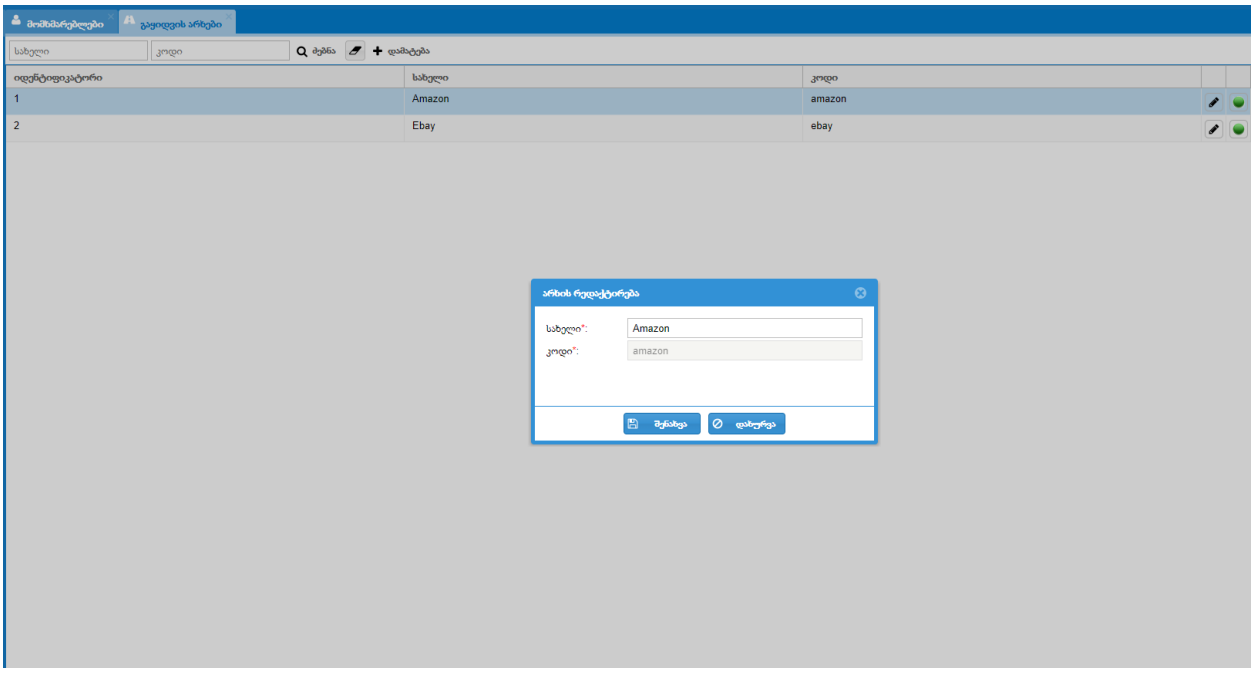
```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns2:getPromoCampaignsResponse
xmlns:ns2="http://pcms.azry.com">
            <ns2:Campaign>
                <ns2:Id>2</ns2:Id>
                <ns2:Name>Amazon Huge Payments</ns2:Name>
                <ns2:StartDate>2019-07-
10T12:57:42+04:00</ns2:StartDate>
                <ns2:EndDate>2019-12-20T00:00:00+04:00</ns2:EndDate>
                <ns2:Priority>HIGH</ns2:Priority>

<ns2:VoucherExpirationTime>5</ns2:VoucherExpirationTime>
                <ns2:MinPaymentAmount>500000</ns2:MinPaymentAmount>
                <ns2:NotPrintReceipt>false</ns2:NotPrintReceipt>
                <ns2:IssuedCodeMaxAmount>0</ns2:IssuedCodeMaxAmount>
                <ns2:ValidatePayParam>false</ns2:ValidatePayParam>
                <ns2:NotSendSms>false</ns2:NotSendSms>
            </ns2:Campaign>
        </ns2:getPromoCampaignsResponse>
    </soap:Body>
</soap:Envelope>
```

4 PCMS ვებ-გვერდი (მართვის პანელი)

პრომო კამპანიების მართვა მთლიანად მორგებულია მომხმარებელზე, ვებ აპლიკაციის საშუალებით მომხმარებლებს შეუძლიათ დააკონფიგურირონ მართვის პარამეტრები, შექმნან საჭირო მონაცემები, ამონიტორინგონ სისტემა და პრობლემების შემთხვევაში გამოასწორონ ისინი. ამ ყველაფრის გაკეთება შესაძლებელია იმ რამდენიმე ფეიჯით რომელიც არის PCMS-ს ვებ აპლიკაციაში.

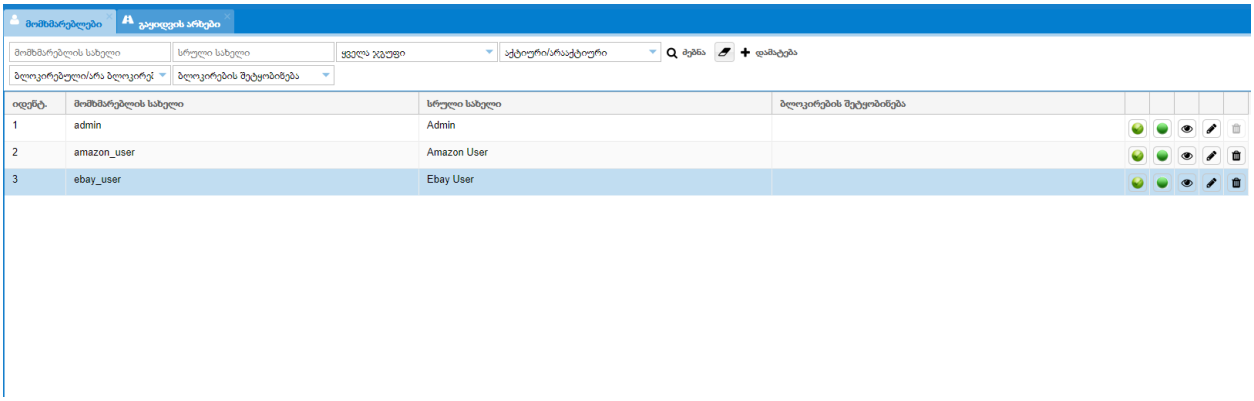
4.1 გაყიდვის არხები

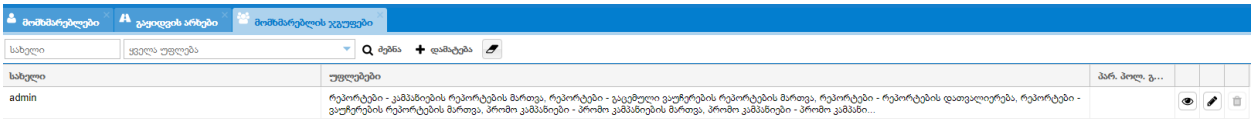
PCMS არის შექმნილი ისე, რომ შესაძლოა ერთი კონკრეტული აპლიკაცია გამოიყენოს რამდენიმე კლიენტმა კომპანიამ. ამისთვის სისტემაში არსებობს არხის ცნება. ყველა სხვა ფუნქციონალი აგებულია არხების მიხედვით, მაგალითად მომხმარებელი ეკუთვნის კონკრეტულ არხს და შეუძლია მხოლოდ მისთვის შექმნილი კამპანიების მართვა. ასევე შესაძლებელია ერთი კამპანია იყოს რამდენიმე არხისთვისაც. ამ ფუნცქიონალით შესაძლებელია რამდენიმე კომპანიამ იმუშაოს ერთ აპლიკაციაზე ისე რომ ერთმანეთთან არანაირი თანაკვეთა არ გააჩნდეთ.



4.2 მომხმარებლები და მომხმარებლის ჯგუფები

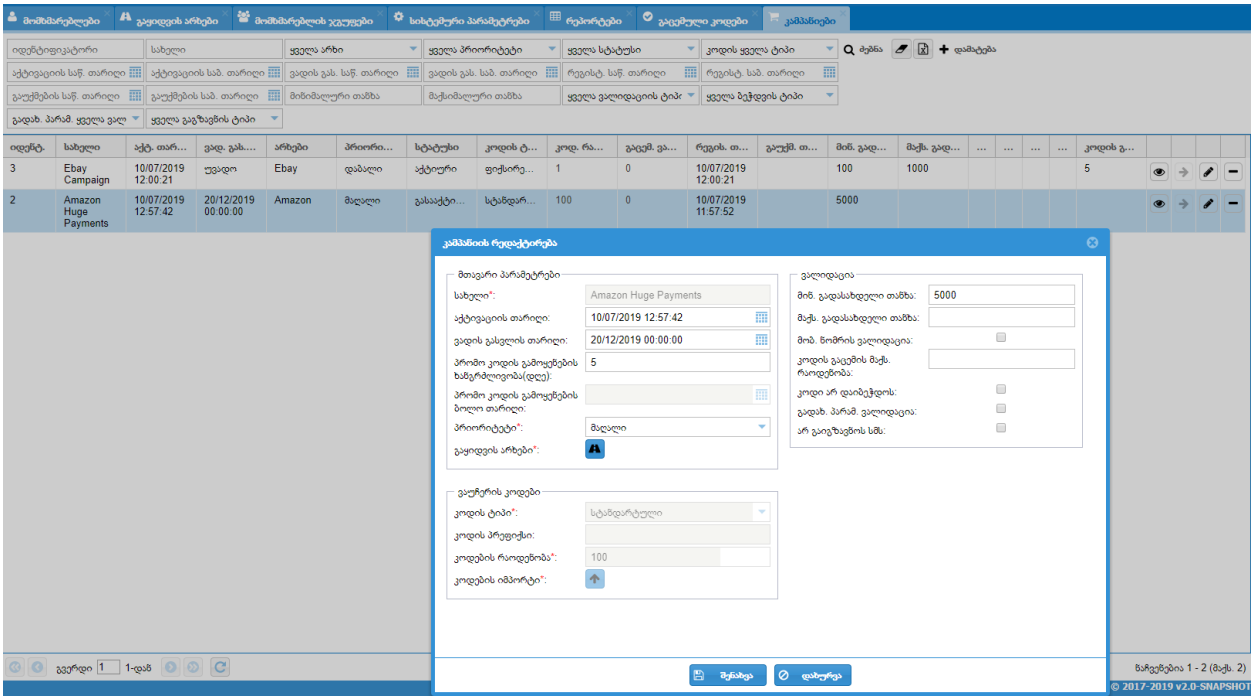
სისტემაში შესვლისთვის და მისი მონიტორინგისთვის აუცილებელია გვყავდეს მომხმარებელი, რომელმაც შესვლისას უნდა გაიაროს ავტორიზაცია. ყველა მომხმარებელი გაწევრიანებულია გარკვეულ მომხმარებლის ჯგუფებში, რომელსაც თავის მხრივ აქვს გარკვეული უფლებები. მომხმარებელის უფლებები განისაზღვრება მისი ჯგუფების უფლებების გაერთიანებით. ეს კონფიგურაცია იძლევა საშუალებას, აპლიკაციას იყენებდეს უამრავი განსხვავებული დანიშნულების მქონე მომხმარებელი, ისე რომ მათ არ შეეძლოთ იმაზე მეტი ინფორმაციის ნახვა, ვიდრე პროტოკოლით არის გაწერილი.





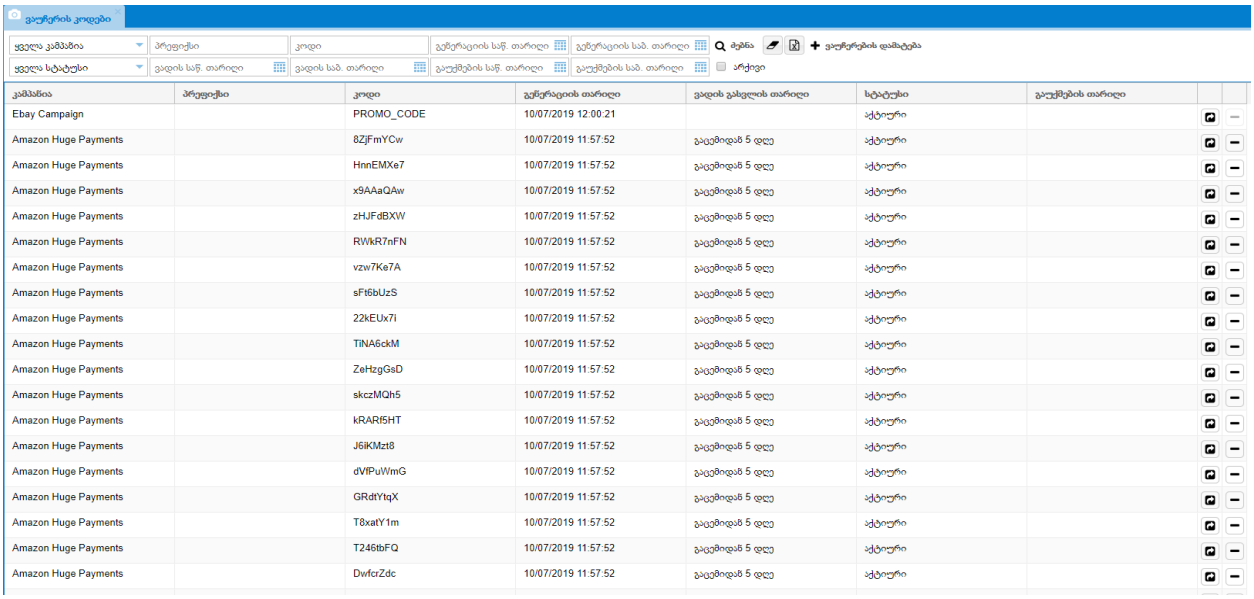
4.3 კამპანიები

აპლიკაციის მთავარი გვერდი არის კამპანიები. სწორედ ამ გვერდზე იქმნდება ახალი კამპანიები, იცვლება ძველები და შესაძლებელია არსებული კამპანიების დათვალიერება. კამპანიის შექმნისას მომხარებელს უწევს დაკვირვებით გაწეროს საკმაოდ დიდი ინფორმაცია, რათა შეიქმნას მისთვის სასურველი კამპანია. მათ შორის აღსანიშნავია ძირითადი პარამეტრები და ვალიდაციები. ვალიდაცია პრაქტიკულად არის კამპანიის გამოყენების ინსტრუქცია, ის განაგებს თუ რა კონკრეტულ მდგომარეობაში გაიცემა კამპანიის ვაუჩერი.



4.4 ვაუჩერის კოდები

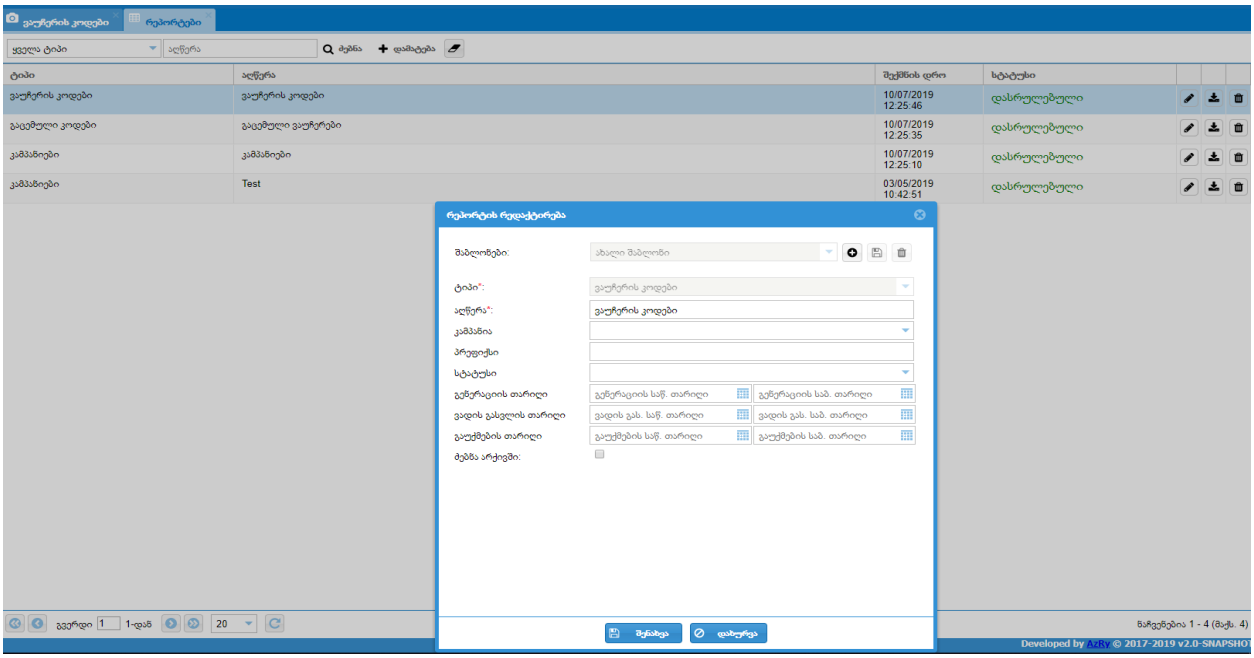
არსებული ვაუჩერების ნახვა შესაძლებელია ვაუჩერის კოდების გვერდიდან. აქ ჩანს თუ რომელი ვაუჩერებია სისტემაში და მათი დაწვრილებითი ინფორმაცია. გვერდზე ასევე არის რამდენიმე ქმედება. მათ შორის გამოსაყოფია ვაუჩერის ხელით გაცემა და გაუქმება. ვაუჩერის ხელით გაცემა არის საკმაოდ გამოყენებადი ფუნქციონალი. თუ გარკვეული შეფერხების გამო კლიენტმა ვერ მიიღო ვაუჩერი, შესაძლებელია მისი მოგვიანებით ხელით გაცემა. ხელით გაცემის დროს გამოდის მობილური ნომრის შესაყვანი ფანჯარა და ვაუჩერის კოდი იგზავნება მითითებულ ნომერზე.



4.5 რეპორტები

სისტემის მონიტორინგისთვის არსებობს რეპორტების გვერდი. რეპორტინგი არის მონაცემების ექსელში ასახვის პროცესი. რა თქმა უნდა ეს არის აპლიკაციის ერთ-ერთი მთავარი ფუნქციონალი, რადგან გარკვეულ პერიოდში ერთხელ აუცილებლად იქმნდება საჭიროება, შეგროვდეს და გაანალიზდეს მიღებული შედეგები. სისტემაში არსებობს შემდეგი სახის რეპორტები:

1) კამპანიები
2) ვაუჩერის კოდები
3) გაცემული ვაუჩერები



5 სისტემის სანდოობა

კომპიუტერული სისტემის ერთ-ერთი მნიშვნელოვანი თვისება არის სანდოობა. სისტემის სანდოობა ასახავს მისი მომხმარებლის მისდამი ნდობის ხარისხს. მომხმარებლის რწმენის ხარისხს, რომ სისტემა იმუშავებს მოლოდინის შესაბამისად და არ განიცდის ავარიას ნორმალური მუშაობის რეჟიმში.

ზოგადად სისტემის ავარიამ შეიძლება დიდი გავლება იქონიოს ხალხზე, რომელიც დამოკიდებულია სისტემაზე. ასევე საკმაოდ დიდი ნეგატიური გავლენა შეიძლება იქონიოს აღნიშნული სისტემის ავარიამ როგორც კომპანიის იმიჯზე ასევე არის ფინანსური ზარალის დიდი რისკი. შესაბამისად პროექტზე მუშაობისას ძალიან დიდი ყურადღება მიექცა სისტემის სანდოობას. სანდოობის ძირითადი თვისებებია

- ხელმისაწვდომობა
- საიმედოობა
- უსაფრთხოება
- დაცულობა

5.1 ხელმისაწვდომობა და საიმედოობა

ხელმისაწვდომობა არის ალბათობა იმისა, რომ დროის მოცემულ მომენტში სისტემა იქნება ხელმისაწვდომი, მუშა მდგომარეობაში და შეძლებს სერვისებით მომხმარებლის უზრუნველყოფას, ხოლო საიმედოობა მოცემულ გარემოში, გარკვეული დროის განმავლობაში სისტემის უშეცდომო მუშაობის ალბათობა.

ორივე თვისება ერთმანეთთან კავშირშია და შეიძლება გამოსახულ იქნას რიცხობრივად. მაგალითად 0.999 ხელმისაწვდომობა ნიშნავს, რომ სისტემა მუშა მდგომარეობაშია დროის 99.9%-ში.

ამ ფაქტორებით PCMS შეიძლება შეფასდეს დაახლოებით 99%-თ. პრაქტიკულად დღის სრული პერიოდის მანძილზე გარანტირებულია პროგრამული უზრუნველყოფის ხელმისაწვდომობა და საიმედოობა. ამ შედეგის მისაღწევად სისტემაში დროთა განმავლობაში ჩაიდო მრავალი დამატებითი ფუქცნიონალი, რომელიც უზრუნველყოფს ავარიების მინიმალიზაციას.

მიუხედავად ასეთი მაღალი შეფასებიდა, ყოველთვის არსებობს ავარიის ან გაუმართავად მუშაობის ალბათობა. სწორედ ასეთი შემთხვევებისთვის სისტემაში ჩადებულია რამდენიმე ხელით ქმედების საშუალება, რომელიც გამოასწორებს ავარიის დროს გამოწვებულ ხარვეზებს.

- **ვაუჩერის ხელით გაცემა** - შესაძლებელია რაიმე გაუთვალისწინებელი შეცდომის გამო(საყურადღებოა, რომ ეს ხარვეზი შეიძლება გამოწვეული იყოს როგორც სისტემის

გაუმართაობით, ასევე მომსახურე პერსონალის ადამიანური შეცდომით) კლიენტმა ვერ მიიღოს დამსახურებული ვაუჩერი. ამის გამოსასწორებლად სისტემაში არსებობს ვაუჩერის ხელით გაცემის ფუნქციონალი. შესაბამისი უფლების ქმონე ადამიანს შეუძლია კონკრეტული ვაუჩერი გაცეს სისტემიდან, მიუთითოს კლიენტის საკონტაქტო ინფორმაცია და გაიგზავნე შესაბამისი შეტყობინება.

- **ვაუჩერის გაუქმება** - ასევე რაიმე გაუთვალისწინებელი შეცდომის გამო შესაძლებელია კონკრეტული ვაუჩერი გათამაშდეს არასწორად ან თუნდაც გათამაშდეს რაიმე წესების დარღვევით. ასეთ სიტუაციაში შესაძლებელია პრომო კოდის გაუქმება და მას ვეღარ გამოიყენებს ვერავინ.
- **შეტყობინების თავიდან გაგზავნა** - შეტყობინების თავიდან გაგზავნა გამოიყენება ისეთ სიტუაციაში, როცა კლიენტს არ მიუღია შეტყობინება ან შემთხვევით მიუთითა არასწორი საკონტაქტო ნომერი. ასეთ შემთხვევაში შესაძლებელია შეტყობინების თავიდან გაგზავნა როგორც მითითებულ ნომერზე, ასევე დამატებით სხვა ნებისმიერ საკონტაქტო ნომერზე.

5.2 დაცულობა და უსაფრთხოება

სისტემის დაცულობა ასახავს სისტემის შესაძლებლობას, თავი დაიცვას გამიზნული თუ შემთხვევითი თავდასხმებისგან. დაცულობა განსაკუთრებით მნიშვნელოვანია იმ სისტემებისთვის, რომლებიც ლოკალურ და გლობალურ ქსელთან არიან მიერთებულნი და შესაძლებელია მათზე გარე თავდასხმის განხორციელება.

უსაფრთხოება არის სისტემის თვისება, რომელიც ასახავს მის შესაძლებლობას, იმუშაოს ნორმალურ და არანორმალურ რეჟიმში იმგვარად, რომ ზიანი არ მიაყენოს მის მომხმარებელს და გარემოს. პროგრამული უზრუნველყოფის უსაფრთხოების განხილვა მნიშვნელოვანია, რადგან დღეს არსებული კრიტიკული სისტემების მართვა სწორედ პროგრამული უზრუნველყოფით ხდება.

PCMS-ს დაცულობასა და უსაფრთხოებისთვის დეველოპმენტის დროს გაკეთებულია რამდენიმე მნიშვნელოვანი საკითხი.

- აპლიკაციაში კავშირი მთლიანად აწყობილია https პროტოკოლით. როგორც ვებ სერვისები, ასევე საიტზე წვდომა ხორციელდება https-თ. ამისთვის შექმნილია სანდო სერთიფიკატი, რომელიც დამატებულია სისტემის სერვერულ ნაწილში და მისი დახმარებით ხდება ქსელში მოძრავი ყველანაირი ინფორმაციის დაშიფვრა.

- ვებ სერვისის უსაფრთხოებისთვის გამოყენებულია ე.წ. **basic** ავტორიზაცია. ეს არ გადის ჩვეულებრივი მომხმარებლების ავტორიზაციაზე, არამედ დამატებულია ახალი Login მოდული, რაც იძლევა ვებ სერვისებისა და აპლიკაციაში არსებული სხვა ნაწილების ერთმანეთთან სრულ დამოუკიდებლობის გარანტიას.

- მომხმარებლის პაროლების „სიძლიერის“ არჩევის თუ პერიოდული ცვლილებისთვის შემუშავებულია სპეციალური პაროლების პოლიტიკა, რომლის გამოც მაქსიმალურადაა შემცირებული მომხმარებლის პაროლის გავრცელების ალბათობა.

- სისტემის სხვადასხვა კომპონენტებთან კავშირისას (მაგლითად: მონაცემთა ბაზა, საჭირო სერვისები და ა.შ.) გამოყენებული პაროლები კონფიგურაციის ფაილებში ინახება დაშიფრულად და არა ღია ტექსტით, რაც უზრუნველყოფს ამ პაროლების ნებისმიერი, მათ შორის პრივილიგირებული მომხმარებლებისგან დამალვას.

სისტემის უსაფრთხოების ძირეული ნაწილი დაკისრებულია JBOSS აპლიკაციის სერვერზე. JBOSS გააჩნია ჩაშენებული თვისებები, რომელიც ქმნის სისტემის სრულ უსაფრთხოებას და იძლევა სანდოობის გარანტიას.

6 დასკვნა

სამაგისტრო ნაშრომში განხილულია პრომო-კამპანიების მართვის სისტემის პროგრამული უზრუნველყოფის შექმნის, სისტემის მოდულირების და რეალურ გარემოში მუშაობის სტატისტიკური ანალიზი. სისტემა უზრუნველყოფს პრომო-კამპანიების (წამახალისებელი გათამაშებების) შექმნას, მონიტორინგს, კლიენტებთან მიწოდებას და დაფიქსირებული სტატისტიკის შეგროვებას. სისტემა ისე არის აგებული, რომ მისი მუშაობის პერიოდში მაქსიმალური სიზუსტით, სისწრაფით და საიმედოობით მოემსახუროს მრავალრიცხოვან კლიენტებს. ზემოთ განხილულია რამდენიმე სახის პრობლემა, რომელიც წარმოიშვა პროგრამულ უზრუნველყოფაზე მუშაობის დროს და გადაიჭრა ისე, რომ არ დარღვეულიყო მთლიანობა და სისტემა დარჩენილიყო ხარისხის მაღალ საფეხურზე.

ნაშრომში აღწერილი ყველა ასპექტი დაფუძნებულია რეალურ გარემოში არსებულ ფაქტებზე. სისტემის დეველოპმენტი მიმდინარეობდა უწყვეტ რეჟიმში და გაგრძელდა რეალურ გარემოში დანერგვის შემდეგ, საჭირო ცვლილებების აღმოჩენით და პრობლემების გადაჭრით. პროგრამული უზრუნველყოფა დღესაც დანერგილია ერთ-ერთ წამყვან საბანკო კომპანიაში, მუშაობს განსაკუთრებული შეფერხების გარეშე და ემსახურება რამდენიმე ათას კლიენტს.

7 გამოყენებული ლიტერატურა

[1] Bruce Eckel President, MindView, Inc., Thinking in Java, Fourth Edition, January 2006;
[2] Andrew Lee Rubinger & Bill Burke, Enterprise Java Beans 3.1, O'Reilly, 6th Edition, Beijing 2010;
[3] Markus Eisele, Modern Java EE Design Patterns, O'Reilly, Beijing 2016;
[4] Mike Keith and Merrick Schnicariol, Pro JPA 2, Mastering the Java Persistence API, Apress 2016;
[5] Adam Tacy, Robert Hanson, Jason Essigton, Anna Tokke, GWT In Action Second Addition, Manning, Second Edition, 2013;
[6] Ramez Elmasi, Shamkart B. Navathe, Fundamentals of database systems seventh edition, Elmasri Navathe, 7th Edition, 2016;
[7] Javid Jamae, Peter Johnson, JBoss in action, 2015;
[8] Brian Suda, SOAP Web Services, 2010;
[9] kevin Zhang, Design Patterns, Elements of Reusable Object-oriented Software, Beijing 2017;
[10] Jeanne Boyarsky and Scott Selikoff, Oracle Certified Professional Java SE 8 Programmer II Study Guide, 2015;
[11] Linda Demichiel, Bill Shannon, Java Platform, Enterprise Edition Specification, 2012;