МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Інститут післядипломної освіти

Кваліфікаційна робота бакалавра

на тему: Розробка web-сервісу служби продажу квитків

Виконав студент групи КН-5
спеціальності 122 Комп'ютерні науки
Сербін Сергій Валерійович

Керівник к.т.н., доцент
Гнатовська Ганна Арнольдівна

Консультант

Рецензент к.т.н., доцент
Терещенко Тетяна Михайлівна

Одеса 2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Хостинг – послуга з надання простору для розміщення сайтів в Інтернеті.

CRUD – (Creation, Retrieval, Update, Destruction) – шаблон проектування

DCE – (Distributed Computing Environment) – розподілена обчислювальна середовище

DCOM – (Distributed Common Object Model) – розподілена об'єктна модель компонентів

SADT (Structured Analysis and Design Technique) – методологія структурного аналізу і проектування

SOAP – (Simple Object Access Protocol) – протокол доступу до об'єктів

SOA – (Service Oriented Architecture) – сервіс-орієнтована архітектура

UDDI – (Universal Description Discovery & Integration) – інструмент для розташування описів web-сервісів

WSDL – (Web Services Description Language) – мова опису web-сервісів

ВСТУП

У сучасному світі розвитку інформаційних технологій, мережа Інтернет стала загальновизнаним фактором ділового і громадського життя. Широка поширеність, пропускна здатність, яка зростає створюють умови, при яких вигідно та зручно вирішувати багато завдань за допомогою саме Інтернет-технологій. Інтернет об'єднує в собі багато різних платформ, а інформація міститься в різноманітних джерелах даних, тому проблема зв'язку таких різнорідних даних, а також створення засобу, який дозволяє отримувати їх у вигляді зручному для подальшої обробки є актуальним завданням. Одним з найбільш актуальних напрямків розвитку комп'ютерних технологій є розробка спеціалізованих інформаційних систем, використання яких є потужним засобом підвищення ефективності роботи будь-якого підприємства, організації, установи [1].

Розвиток розподілених інформаційних систем, об'єктно-реляційних баз даних, веб-орієнтованих засобів фінансової діяльності підприємств впливає на методологію і технологію організації діяльності підприємств та організацій і передбачає реалізацію нового підходу до її інформаційного забезпечення. Сучасна комерційна діяльність підприємств та організацій вимагає високих швидкостей обміну інформацією. Від того наскільки швидко надійде інформація про вихід на ринок тих чи інших товарів, наскільки ця інформація буде яскравою і доступною залежить прибуток підприємства. Збір, зберігання, доступ до отриманих даних, і т.д. вимагає значних витрат, а отже, ефективного використання часового ресурсу, для чого необхідна правильна організація всього процесу роботи з комерційною інформацією [2].

Стрімкий розвиток глобальної мережі Інтернет збільшив потреби у появі послуг, що надаються в реальному часі. На сьогоднішній день існує безліч сервісів, що надають послуги з обміну інформацією в мережі Internet, які стали невід'ємною і важливою частиною ефективності роботи багатьох

фірм, підприємств і організацій, а також приватних осіб. Мережа Internet стала основним джерелом оперативного отримання інформації.

Сучасні інформаційні системи характеризуються своєю розподіленою структурою. Web-сервіс – це мережна технологія, що забезпечує взаємодію між різними додатками на основі веб-стандартів. Web-сервіси дають можливість звернутися з одного додатку до іншого і виконувати певні функції. Серед переваг застосування таких систем можна виділити наступні: скорочення часу розробки, скорочення кількості помилок, повторне використання програмних компонентів, полегшується майбутня зміна системи. Розподілене виконання забезпечує передачу даних засобами мережі Internet між філіями підприємств чи організацій, в рамках однієї програми. Традиційний підхід розробки розподілених додатків, підтримуваний технологіями розподілених об'єктів, ґрунтується на тісному зв'язку між усіма програмними компонентами. Слабка зв’язаність програмних компонентів, підтримується технологією веб-сервісів, що дозволяє значно спростити координацію розподілених систем і їх ре конфігурацію. Тому розробка web-сервісу служби продажу квитків, що ґрунтується на використання технології веб-сервісів є актуальною задачею.

Мета дипломної роботи – здійснення проектування та програмної реалізації web-сервісу служби продажу квитків для розповсюдження квитків на різні вистави, концерти, виставки конференції, тощо.

Розробка web-сервісу служби продажу квитків та розміщення сервісу у будь-якій інформаційній системі мережі Інтернет забезпечить можливість залучення потенційних замовників або зацікавлених осіб, що забезпечить можливість здійснення замовлення квитків на різні заходи.

Дипломна робота містіть: 71 сторінка, 11 таблиць, 38 рисунків, 12 посилань та 8 листів Додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ВИЗНАЧЕННЯ ВИМОГ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної області

Стрімкий розвиток Internet-технологій відкриває користувачам нові способи ведення справ, створює безпрецедентні можливості підтримання ділових відносин у віртуальному інформаційному просторі на різноманітних рівнях. Глобальні інформаційні ресурси, зосереджені у світовій мережі Internet нарівні з інформаційними службами нового покоління, можна розглядати як прототип нової системи соціальної комунікації в рамках усієї світової спільноти. Глобальна мережа Internet сьогодні використовується не тільки для обміну інформаційними повідомленнями і для доступу до різноманітних інформаційних ресурсів. Все більше застосування знаходять Internet-технології для здійснення конкретних комерційних операцій. У даному контексті використання Internet і сучасних технологій інформаційного обміну дозволить отримати істотні переваги завдяки скороченню витрат, прискоренню різноманітних бізнес-процесів, внаслідок чого зросте інформаційна привабливість, загальна прибутковість діяльності компаній [3].

Процес використання Internet-технологій для оптимізації інформаційних і комерційних процедур, що відбуваються в рамках глобальних процесів соціальної комунікації, обумовив появу цілого ряду нових способів отримання прибутку за допомогою мережі Internet[4]. Якісний інформаційний ресурс комерційного підприємства забезпечує ефективне ведення маркетингової та ринкової політики. На технічному рівні – це гарний зміст, хороша структура, привабливий дизайн. Хороший інформаційний ресурс швидко завантажуватися, не напружуючи користувача. Він повинен мати просту і прозору структуру, в якій легко орієнтуватися користувачу.

Для web-сервісу служби продажу квитків, що підтримує функції замовлення квитків, досить важливий привабливий дизайн. Інформаційна система, до якої буде додано веб-сервіс, що підтримує функції замовлення квитків, повинен бути ефективним засобом залучення клієнтів для здійснення продажів засобами мережі Інтернет. Інтерфейс сервісу має бути продуманим та орієнтованим на інтереси перш за все клієнтів. Клієнт повинен отримувати на запропонованому Інтернет-ресурсі всю необхідну йому інформацію.

Важливою складовою web-сервісу служби продажу квитків є реклама культурно-масових заходів – вистав, конференцій, відеопереглядів, виставок, презентацій, концертів і т.п. Наведена інформація повинна становити інтерес, буду зручною, наявної не тільки для клієнта – головного користувача сервісу, а й для організаторів цих заходів. Інформаційна система, що включає web-сервіс служби продажу квитків знаходиться у віртуальному просторі мережі Інтернет, який підтримує функції замовлення квитків і надає клієнтам послуги, сама по собі прибутку не приносить. Але навіть найефективніша електронна система збору замовлень не ефективна, якщо вона, в свою чергу, не спирається на продуману систему оплати та доставки замовлень. Недоліки в цих питаннях впливають на престиж та прибутки підприємств, які надають такі послуги.

1.2 Опис та визначення функцій сервісу

Можливості сучасного розвитку мережі Internet визначають вимоги до застосування сучасних інформаційних технологій для здійснення інформування комерційним підприємством про свою діяльність та залучення клієнтів засобами мережі. Для того щоб розширити свої можливості і коло клієнтів і, відповідно, збільшити свої доходи, підприємства, які здійснюють діяльність з організації культурних заходів потребує створення як інформаційної так і рекламної веб-системи у мережі Internet, яка буде

підвищувати обізнаність клієнтів про діяльність установи, про майбутні заходи в культурному житті та надавати клієнтам можливість безпосереднього замовлення квитків на цікаві для них культурно-масові заходи. Аналіз системи маркетологами підприємства може надавати інформацію що до популярності серед глядачів того чи іншого культурно-масового заходу. Підприємство за цими даними може здійснювати формування споживчої пропозиції, що позначитися на отриманні підприємством прибутку.

Квитки на культурно-масові заходи можуть поширюватися як шляхом продажу їх в касах закладів (не обов'язково саме в тих, де даний захід буде проводитися), так і за допомогою агентів-розповсюджувачів, що мають у своєму розпорядженні мобільні засоби зв'язку (планшети, смартфони, тощо за допомогою яких вони можуть здійснювати доступ в Інтернет), а також клієнти зможуть забронювати квитки безпосередньо на сайті фірми-організатора заходів з подальшим їх отриманням в касі закладу.

Web-сервіс служби продажу квитків на культурно-масові заходи повинен надавати користувачам наступні можливості:

- інформування потенційних клієнтів про майбутні заходи;
- поширення квитків на заходи за допомогою мережі Internet і за допомогою кас, які перебувають у закладах, де будуть проводитися заходи, але система повинна мати розподілену структуру, тобто повинна підтримуватися можливість придбання квитків в будь-якій касі будь-якого закладу, де буде відкритий доступ до системи продажу квитків організатора;
- повинна підтримуватися можливість передоплати квитків шляхом нарахування бонусів та ведення особистого рахунку користувача з урахуванням можливості поповнення рахунку;
- для адміністратора системи повинна підтримуватися можливість додавання інформації про заклади (концертні зали, театри) і про майбутні заходи;

- для адміністратора сервісу повинна підтримуватися можливість опису плану концертного (виставкового) залу для універсальності сервісу;
- для касира повинна підтримуватися можливість пошуку замовлень, їх редагування, оброблення;
- повинна підтримуватися можливість для касира поповнення клієнтських рахунків;
- клієнти системи повинні поділятися на звичайних користувачів і привілейованих(зареєстрованих). Звичайний користувач повинен переходити в розряд привілейованих після заповнення реєстраційної форми та подальшого отримання свого особистого коду і можливості накопичення бонусів;
- звичайні (незареєстровані) користувачі повинні мати можливість перегляду інформації про заходи та в яких саме закладах вони будуть відбуватися, а також можливість замовлення квитка з подальшою оплатою замовлення готівкою в касі;
- зареєстровані користувачі повинні мати такі ж права, як і звичайні користувачі, за винятком того, що вони повинні мати можливість оплати замовлених квитків за допомогою бонусів.

1.3 Постановка завдання

Метою дипломної роботи є розробка web-сервісу служби продажу квитків, який засновано на використанні сервіс-орієнтованої технології. Для здійснення всіх етапів розробки сервісу, необхідно вирішити наступні завдання:

- провести дослідження предметної області;
- визначити функції користувачів;
- провести моделювання бізнес-процесів визначеної предметної області;

– сформулювати вимоги до функцій створюваного сервісу;
– провести проектування функціональної моделі взаємодії компонентів проектованої системи;
– провести вибір апаратно-програмної платформи для реалізації сервісу;
– провести вибір та обґрунтування архітектурних рішень та засобів для реалізації сервісу;
– здійснити програмну реалізацію сервісу.

Застосування у будь-якій розподіленої інформаційної системи сервісу служби продажу квитків, забезпечить інформаційну присутність організації, що організує заходи у мережі Інтернет, забезпечить залучення та інформування клієнтів про майбутні різні культурні заходи, а також забезпечить можливість безпосереднього замовлення квитків.

Web-сервіс служби продажу квитків повинен підтримувати 4 категорії користувачів:

– гість – незареєстрований користувач;
– привілейований (зареєстрований) користувач – це зареєстрований користувач з можливістю ведення особистого рахунку та накопичення бонусів;
– адміністратор – безпосередній адміністратор системи;
– касир – користувач, який нараховує бонуси і проводить облік та управління замовленнями та продаж квитків.

В рамках даної роботи потрібно створити веб-сервіс, з функціями, які призначені для здійснення розповсюдження квитків установою, що організує культурні заходи, а також інформує клієнтів про майбутні культурно-масові заходи. Реалізований веб-сервіс повин забезпечувати такі функції:

– виведення анонсів заходів;
– замовлення квитків;
– оформлення продажу квитків (для касира);
– управління інформацією в системі (для адміністратора).

2 ВИБІР АРХІТЕКТУРИ ТА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ WEB-СЕРВІСУ

2.1 Застосування сервіс-орієнтованої архітектури web-сервісу

Для здійснення програмної реалізації web-сервісу служби продажу квитків було обрано сервіс-орієнтовану архітектуру (SOA, service-oriented architecture), яка являє собою модульний підхід до розробки програмного забезпечення, заснований на використанні сервісів (служб) зі стандартизованими інтерфейсами.

Інтеграція програмних засобів – завдання, можливо, найбільш складне, оскільки саме програмні засоби забезпечують інтеграцію всіх інших складових системи.

Підприємство має в максимально використовувати наявні активи інформаційних технологій, щоб знизити витрати і зменшити терміни впровадження. Також при розробці нових або модифікації наявних додатків доцільно максимально використовувати вже наявні компоненти. При інтеграції же з інформаційними системами партнерів, постачальників, споживачів і т.п. приведення всіх компонентів системи до єдиній платформі просто неможливо. Необхідність вирішення цих проблем породила концепцію сервісно-орієнтованої архітектури. Програмні засоби, що реалізують компоненти бізнес-процесів, оформляються як сервіси. Сервіс – це програмний компонент, який реалізує закінчену функцію надання або обробки даних, переводячи їх з одного цілісного стану в інший цілісний. Основною відмінністю сервісу від звичайного компонента є стандартний і переносний незалежний інтерфейс. Клієнт, який звертається до сервісу, не зобов'язаний нічого знати про подробиці реалізації сервісу: якою мовою і в якій моделі програмування він створений, на яких апаратних засобах, в якому операційному середовищі, на якій платформі проміжного програмного забезпечення він виконується [4].

Сервісно-орієнтована архітектура, дозволяє компонувати бізнес-процеси з компонентів, що виконуються на різних платформах, представляти у вигляді сервісів і повторно використовувати в нових бізнес-процесах успадковані компоненти. В останньому випадку саме компоненти успадкування не змінюються, але для них тільки робиться оболонка, що адаптує їх інтерфейси до стандартів сервісів. Оформлення процесу як сервісу не пред'являє будь-яких вимог до розробки процесу, а тільки вимоги до його інтерфейсу. Щоб успішно виконувати свої функції, платформа, яка втілює сервісно-орієнтовану архітектуру, повинна відповідати таким вимогам [5]:

– забезпечувати специфікації інтерфейсу, які були б прийняті більшістю розробників компонентів;
– використовувати загальноприйняті протоколи для взаємодії сервісів і клієнтів;
– не використовувати в інтерфейсі будь-які складні або закриті формати представлення інформації;
– не вимагати для своєї підтримки дорогого або ресурсоемкого програмного забезпечення.

В основі SOA лежать принципи багатократного використання функціональних елементів, ліквідації дублювання функціональності в програмне забезпечення, уніфікації типових операційних процесів, забезпечення переведення операційної моделі компанії на централізовані процеси і функціональну організацію на основі промислової платформи інтеграції. Компоненти програми можуть бути розподілені по різних вузлах мережі, і пропонуються як незалежні, слабо пов'язані, замінні сервіси-додатки. Програмні комплекси, розроблені відповідно до SOA, часто реалізуються як набір web-сервіс, інтегрованих за допомогою відомих стандартних протоколів (SOAP, WSDL, і т. п.).

Архітектуру SOA можна застосовувати для ефективної організації взаємодії між додатками корпоративних інформаційних систем. Для цього слід знати, які бізнес-функції необхідні підприємству для здійснення його

діяльності. З цією метою необхідно зробити декомпозицію функціональних блоків бізнес-процесів до отримання окремих бізнес функцій, які можна ототожнити з сервісами. З функціональної точки зору, додаток розпадається на сукупність взаємодіючих між собою сервісів, що, в свою чергу, можна ототожнити з сервісно-орієнтованою архітектурою, в рамках якої всі функції програми є незалежними сервісами з чітко визначеними інтерфейсами, які викликаються в потрібному порядку з метою формування бізнес-процесів. Також веб-сервіси можуть використовуватися в якості будівельних блоків при створенні різних інформаційних веб-систем [5].

Використання web-сервісів як технологічних специфікацій дозволяє перейти від «розширеного підприємства» та бізнесу «реального часу» об'єднуючого підприємство, постачальників, партнерів, клієнтів в єдину систему. Під web-сервісами розуміються програмні системи, які використовують XML як формат даних, стандарти Web Services Description Language (WSDL), Universal Description, Discovery and Integration (UDDI) і Simple Object Access Protocol (SOAP). WSDL – визначає місце розташування сервісу і відображає операції (або методи), що дозволяють звертатися до цього сервісу [3].

SOAP – це простий заснований на XML протокол для опису формату прийнятих і отриманих повідомлень. Він дозволяє програмам обмінюватися інформацією по транспортним протоколам, таким як HTTP. Стандарт UDDI застосовують для створення каталогів доступних сервісів. SOA – це архітектура, призначена для вільного розміщення функціональних модулів, кожен з яких здатний виконувати певні дії. Ці модулі є описують самі себе програмні компоненти, досяжні через мережу. При цьому модулі публікують свої інтерфейси таким чином, що їх використання не вимагає знання використаних у них рішень і технологій. Основні принципи SOA [5]:

- архітектура, яка не прив'язана до якоїсь певної технології;
- незалежність організації системи від використаної обчислювальної платформи та мов програмування;

– використання сервісів, незалежних від конкретних програм, з однаковими інтерфейсами доступу до них;
– організація сервісів, як слабозв'язаних компонентів для побудови систем.

Основу SOA становить взаємодія трьох учасників: постачальника сервісу, споживача сервісу, реєстру сервісів (рис. 2.1).

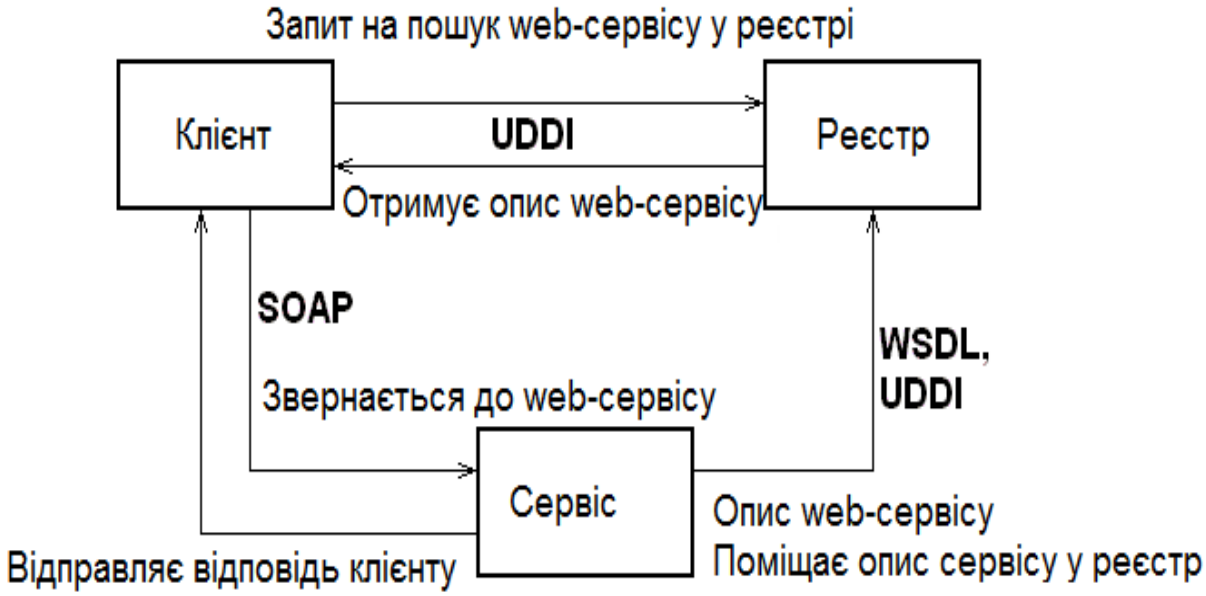


Рисунок 2.1 – Взаємодія складових частин в архітектурі SOA

Ідея сервісів в інформаційних системах як така не нова. Добре відомі такі підходи до її реалізації: Java RMI від компанії Sun Microsystems (Java Remote Method Invocation); CORBA консорціуму Open Management Group (Common Object Request Broker Architecture); DCE запропонована асоціацією Open Group (Distribu ted Computing Environment); Microsoft DCOM (Distributed Component Object Model) [4].

Кожну з архітектур, яка реалізується цими засобами, цілком можна назвати орієнтованої на сервіси, але при цьому кожна з них визначає свої власні формати і протоколи, механізми викликів, інтерфейси для прикладних програм. Недолік універсальності обмежує їх поширення.

2.2 Технології реалізації та функціонування web-сервісу

Web-служба або web-сервіс (англ. Web-service) – це програмна система, що ідентифікується рядком URI, чиї загальнодоступні інтерфейси визначені на мові XML. Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису за допомогою повідомлень, заснованих на XML, і переданих за допомогою інтернет-протоколів. Web-служба є одиницею модульності при використанні сервіс-орієнтованої архітектури додатку.Сервіс – це ресурс, який реалізує якусь функцію з виконання будь-яких дій і має такі властивості [6]:

- є функціонально самостійним об'єктом;
- є повторно використовуваним;
- функціонує незалежно від інших ІС;
- являє собою «чорний ящик» для будь-яких зовнішніх додатків;
- визначається відповідно до певних правил;
- може бути викликаний за допомогою комунікаційних протоколів;
- володіє незалежністю від платформи, на якій він функціонує;
- володіє прозорістю розташування.

Таким чином, з функціональної точки зору, Internet-додаток розпадається на сукупність взаємодіючих між собою web-сервісів.

Оскільки web-сервіс є функціонально самостійним і повторно використовуваним об'єктом, його можна застосовувати неодноразово в різних web-додатків або web-система. Web-сервіс функціонує незалежно від інших ІС, його можна застосовувати в різних незалежних один від одного web-додатків або web-система. Оскільки web-сервіс являє собою «чорний ящик» для будь-яких зовнішніх додатків – все, що повинно бути відомо зовнішнім додаткам – те, що необхідно подати на вхід сервісу і що слід очікувати на виході [7].

В основу web-сервісу покладено ряд технологій і стандартів, і виходячи з цього він визначається відповідно до певних правил. Web-послуги покликані працювати в мережевому середовищі, тому одним з їхніх властивостей є те, що вони можуть бути викликані за допомогою комунікаційних протоколів, а саме – протоколу HTTP. Основу Web-сервіс складають платформенно-незалежні технології World Wide Web (WWW), тому вони володіють незалежністю від ПЗ, на якому реалізовані сервіси, і від платформи, на якій вони функціонують [5].

Користувачеві, який взаємодіє з додатком, що використовують результат роботи web-сервісу, немає необхідності знати про місцезнаходження сервісу – він може перебувати як на тому ж комп'ютері, що і сам додаток, так і на віддаленому сервері. Тому можна стверджувати, що web-послуги володіють прозорістю розташування.

Важлива властивість web-сервісу полягає в тому, що він не залежить від провайдера, комп'ютера або браузера – з його використанням користувач може працювати зі своїми даними в будь-якій точці світу, де у нього є доступ до мережі Internet. Доступ до web-сервісу здійснюється по протоколу HTTP, а обмін даними відбувається в форматі XML (рис. 2.2).

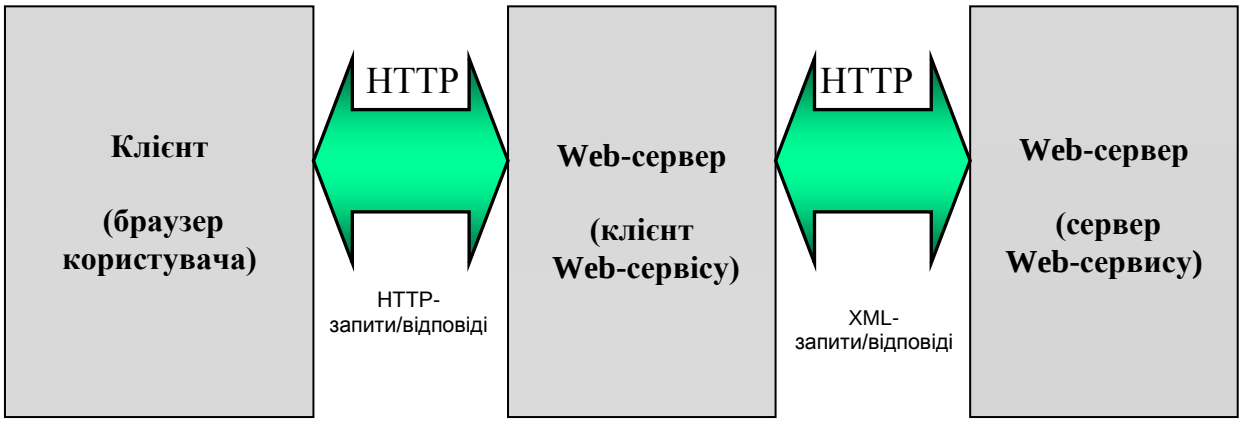


Рисунок 2.2 – Схема роботи web-сервісу

Якщо розділити мережу Інтернет на кілька шарів, можемо виділити, як мінімум, два концептуальних типу додатків – обчислювальні вузли, які реалізують нетривіальні функції та прикладні web-ресурси. При цьому другий, часто зацікавлені в послугах перших. Але й сам Інтернет різнорідний, тобто різні додатки на різних вузлах мережі функціонують на різних апаратно-програмних платформах, і використовують різні технології і мови. Щоб зв'язати все це і надати можливість одним додаткам обмінюватися даними з іншими, і були придумані web-сервіси [5].

По суті, web-сервіси – це реалізація чітких інтерфейсів обміну даними між різними додатками, які написані не тільки на різних мовах, але і розподілені на різних вузлах мережі.

2.3 Визначення протоколів та стандартів взаємодії web-сервісу

В основі web-сервісів лежать Інтернет-стандарти, які визначають протоколи, а не способи їх реалізації. Web-сервіси базуються на трьох основних web-стандартах [7]:

- SOAP (Simple Object Access Protocol) – протокол для посилки повідомлень по протоколу HTTP та іншим Internet -Протокол;
- WSDL (Web Services Description Language) – мова для опису програмних інтерфейсів web-сервісів;
- UDDI (Universal Description, Discovery and Integration) – стандарт для індексації web-сервісів.

Web-сервіси надають інтерфейси для передачі компонентних даних і бізнес-логіки по протоколуHTTP. Для того щоб програми могли використовувати web-сервіси, програмні інтерфейси останніх повинні бути детально описані. Опис може включати таку інформацію, як протокол, адреса сервера, номер використовуваного порту, список доступних операцій, формат запиту і відповіді і т. п. Існуючі web-сервіси описуються в WSDL-документ, які розташовуються або на сервері додатків, або в спеціальних

XML-сховищах. WSDL-документ може посилатися на інші WSDL-документи та документи XSD (XML Schema), в яких описані типи даних, що використовуються web-сервісами. XML-сховище використовуються для управління WSDL-документ. Усередині WSDL-документа знаходиться адресу (URL) web-сервісу. Web-сервіси описані і проіндексовані в бізнес-реєстрі, що містить адреси (URL) WSDL –документів [7].

SOAP (Simple Object Access Protocol) – це стандарт для відсилання та отримання повідомлень по Internet. Специфікація SOAP визначає XML-«конверт» для передачі повідомлень, метод для кодування програмних структур даних у форматі XML, а також засоби зв'язку по протоколу HTTP. SOAP-повідомлення бувають двох типів: запит (Request) і відповідь (Response). Запит викликає метод віддаленого об'єкта, відповідь повертає результат виконання даного методу. UDDI – Universal Description, Discovery and Integration. На рисунку 2.3 наведена схема роботи web-сервісів.

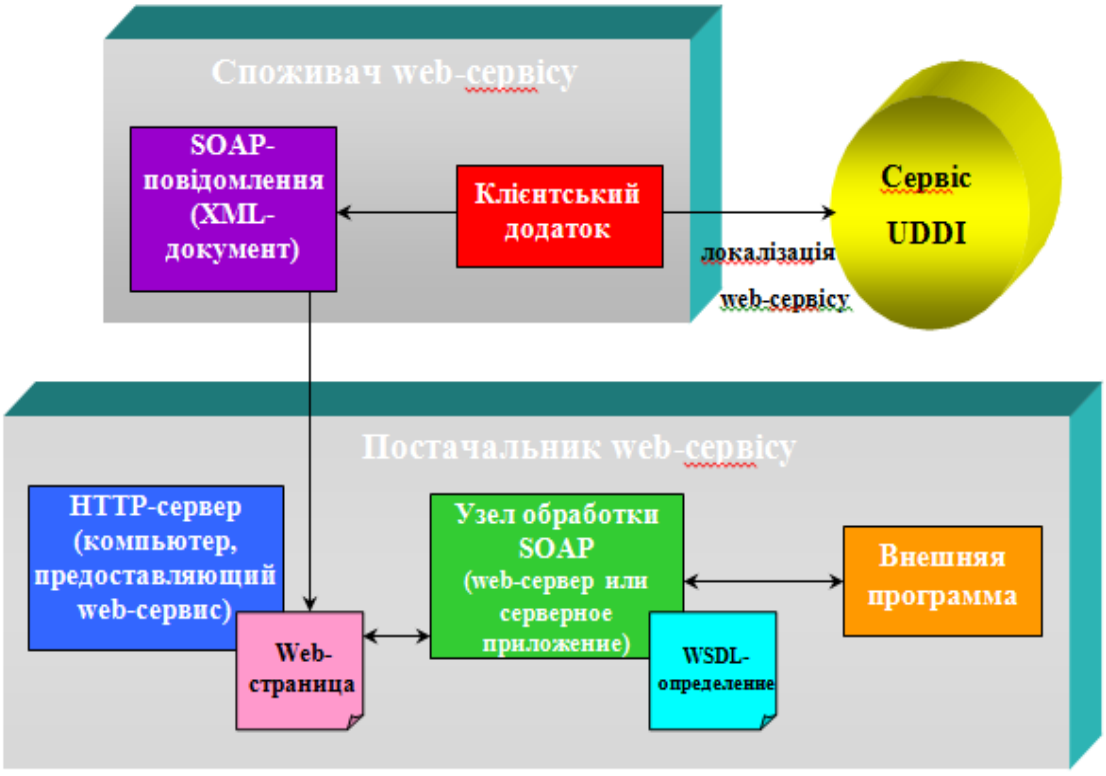


Рисунок 2.3 – Схема роботи web-сервісів

Завдання UDDI – надати механізм для виявлення web-сервісів. UDDI задає бізнес-реєстр, в якому провайдери web-сервісів можуть реєструвати сервіси, а розробники – шукати необхідні їм сервіси. Бізнес-реєстр UDDI сам є SOAP web-сервісом. Він підтримує операції створення, модифікації, видалення та пошуку web-сервісу[6]. Програма-клієнт має знати: які методи сервісу доступні, які параметри потрібні і тип, що повертається результату. Web-сервіс повертає ці дані в SOAP-формат. Для того, щоб передати web-сервіс вказівки зробити що-небудь, ми повинні очевидним чином запакувати наші параметри в XML, причому обов'язково в тому вигляді, в якому web-сервіс зрозуміє. Після цього за допомогою HTTP SOAP переслати ці параметри web-сервісу, і після отримання результату від web-сервісу, теж у вигляді SOAP-пакету, дешифрувати ці SOAP-пакети і відобразити їх в браузері. Кожне SOAP-повідомлення поміщається в «конверт» – SOAP-одиниці обміну повідомленнями: <Soap: Envelope ...</ Soap: Envelope> [5].

SOAP-запит це HTTP POST-запит. У першій сходинці перебуває команда POST. Далі визивается URI сервісу. Крім цього, зазначений Content – Type – це текст у форматі XML. І це дуже важливо, тому що, якщо вказати текст у форматі HTML (text / html), то SOAP-пакет не буде правильно декодований назад.

2.4 Визначення архітектури для реалізації web-сервісів

Основні технології реалізації сервіс-орієнтованого Internet-додатку, заснованого на застосуванні web-сервісу [5]:

- віддалений виклик процедур за допомогою XML – RPC;
- передача стану (Representation State Transfer (REST)) за допомогою RESTful web-сервісів;
- використання Web-сервісу на основі SOAP;

– web-послуги на основі SOAP були розглянуті в попередньому пункті. Це найпопулярніший і широко поширений спосіб реалізації web-сервіс.

XML – RPC. XML-виклик віддалених процедур – це стандарт / протокол виклику віддалених процедур, заснований на XML. Він є прабатьком SOAP, відрізняється винятковою простотою застосування. XML – RPC. В ньому визначаються виклики процедур, закодовані на XML і передані по HTTP. Як і будь-який інший інтерфейс RPC, визначає набір стандартних типів даних і команд, які програміст може використовувати для доступу до функціональності іншої програми, що знаходиться на іншому комп'ютері в мережі. Після декількох циклів по розширенню функціональності, з'явилася система, нині відома як SOAP. Пізніше Майкрософт почала широко рекламувати і впроваджувати SOAP, а початковий XML – RPC був відкинутий компанією-розробником (Microsoft). Але, незважаючи на це, стандарт XML – RPC набув популярності серед програмістів із-за своєї простоти і, за рахунок цього, існує донині. Однак, технологія XML – RPC має ряд істотних недоліків, включаючи дуже примітивний контроль типів даних і відсутність підтримки кодування символів. Оскільки Web-сервіси необов'язково використовувати SOAP, велика група розробників відстоює пропозицію про те, що достатньо просто обмінюватися необробленими XML-документами безпосередньо через HTTP. REST – це стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, який, як правило, використовується для побудови web-сервісів. Системи, що підтримують REST, називаються REST-ful-системами. У загальному випадку REST є дуже простим інтерфейсом управління інформацією без використання якихось додаткових внутрішніх прошарків. Кожна одиниця інформації однозначно визначається глобальним ідентифікатором, таким як URL [7].

Передбачається відмовитися від розробки нових протоколів, а використовувати кілька добре перевірених, вважаючи, що для роботи з

об'єктами цілком достатньо вміти виконувати чотири типи дій: створення (Creation), відновлення (Retrieval), зміна (Update) і знищення (Destruction). З цих дій виходить так званий «шаблон проектування» CRUD. Протокол HTTP визначає методи GET / PUT / POST / DELETE, які й реалізують шаблон CRUD. Прихильники застосування цього стилю для Web-сервісів стверджують, що SOAP складний, обмежує свою користь навантаження XML і не використовує в достатній мірі сильні сторони Web.

Також до REST відносять сервіси, побудовані в цьому стилі, до другого покоління. Решта Web-сервісів вони оголошують сервісами першого покоління, підкреслюючи при цьому, що SOAP є ні що інше, як DCOM або CORBA, але працюють через Internet. Дійсно, спочатку технології, подібні SOAP, називали Web-брокерами або об'єктними брокерами, побудованими в Web. Використовувана в них модель RPC призначена для замкнутого світу, де ви знаєте кожного користувача, де дані можна перерозподіляти між ними, де можливі прямі комунікації. Для успішної роботи в Web необхідно підтримувати складний механізм, що забезпечує взаємодію на випадок внесення системних змін.

Web-сервіси являють собою технологію інтеграції додатків, яка може використовуватися в інформаційних системах, побудованих з використанням технологій Internet. Web-сервіс не є продуктом для кінцевого користувача. Він являє собою заснований на компонентах додаток, дозволяючи багаторазово використовувати свою функціональність в різних середовищах і на клієнтах різних типів. Користувачем web-сервісу завжди є інший додаток. Web-служби можуть використовуватися для вирішення таких завдань [6]:

– з метою здійснення business – to – business транзакцій або з'єднання внутрішніх систем окремих компаній. Web-послуги засновані на використанні загальноприйнятих, відкритих технологій Internet, що не залежать від постачальників ПЗ, за допомогою чого досягається головна перевага web-сервісу як концепції побудови розподілених ІС – їх універсальність. Таким чином, розподілені додатки, побудовані з

використанням технології web-сервісу, забезпечують інтеграцію різнорідних додатків різних учасників єдиного бізнес-процесу. Подібна інтеграція називається додатками B2B. На сьогоднішній день це найбільш широко поширені завдання, які вирішуються із застосуванням web-сервісів. Web-сервіси дозволяють спільно використовувати інформацію або можуть інтегруватися з іншими службами. Наприклад, компанія, що займається електронною комерцією, може звертатися до web-служби для здійснення автоматичного взаємодії з постачальниками;

– в якості готових модулів для розробників. Припустимо, не залежний розробник спроектував web-службу аутентифікації, призначену для застосування в середовищі ASP. NET. Якщо ви побажаєте скористатися цією службою, то за відповідну плату можете придбати місячну підписку на неї. Однак даний процес буде абсолютно прозорим для кінцевого користувача, який вирішить, що зазначені кошти аутентифікації є частиною вашої програми. Такі готові компоненти можна використовувати в web-додатках, а також в настільних і мобільних програмах;
– в якості компонентних бібліотек DLL для багаторазового використання коду. Найпростіший спосіб багаторазового використання певних функціональних можливостей в Інтернет-додатках полягає в проектуванні web-сервісу, до якого могли б звертатися різні клієнти, в тому числі настільні додатки, PDA і браузери. При цьому не важливо, де розташовуються web-служби і клієнти, необхідно лише наявність Інтернет-з'єднання між клієнтом і службою;
– в якості інструментів, що забезпечують взаємодію різних програм в рамках однієї компанії. Web-служби можна використовувати для з'єднання спеціалізованої програми по роботі з платіжними

відомостями з бухгалтерським програмним забезпеченням через захищену корпоративну мережу (а не через Інтернет).

Веб-сервіси позиціонуються як програмне забезпечення проміжного шару. Використовувати веб-сервіси можуть як клієнтські програми, які безпосередньо працюють з користувачем, так і інші додатки. Веб-сервіси розміщуються на серверах додатків. Розробники концепції веб-сервісів пропонують наступні сценарії застосування веб-сервісів. Веб-сервіси, як реалізація логіки додатка (бізнес-логіки) (рис. 2.4).

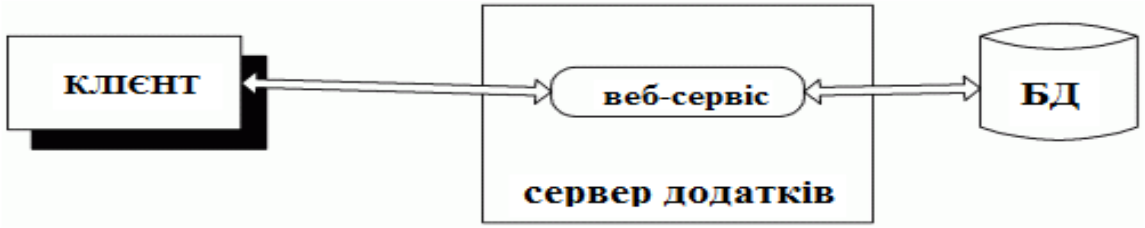


Рисунок 2.4 – Схема застосування веб-сервісу для реалізаціїя бізнес-логіки

Веб-сервіси, як засіб інтеграції. Тобто, використання веб-сервісу як способу доступу віддалених клієнтів до внутрішньої ІС компанії, або для організації взаємодії компонента з різними віддаленими клієнтами (рис. 2.5).

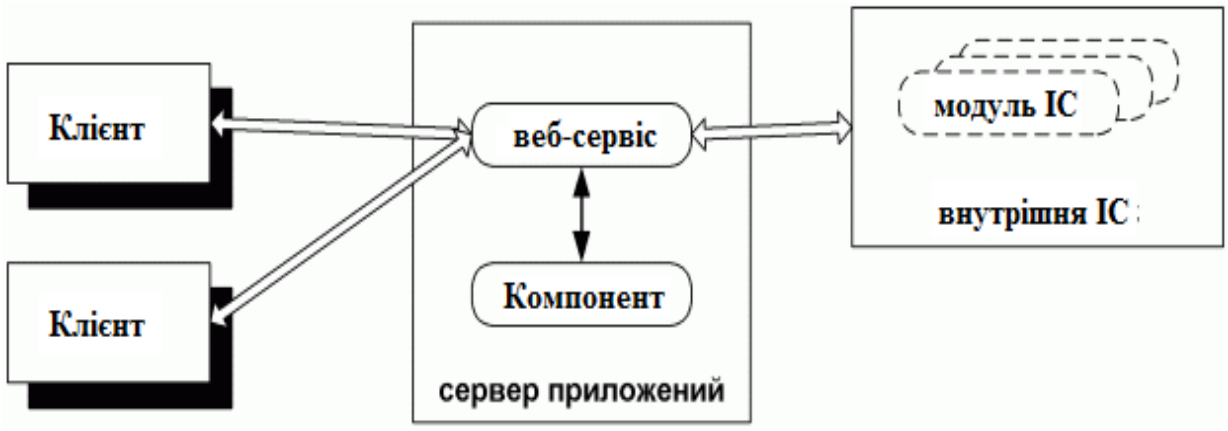


Рисунок 2.5 – Схема застосування веб-сервісу, як засобу інтеграції

Створення альтернативного сервісу. В цьому випадку, при розробці нового веб-сервісу використовується опис інтерфейсу вже існуючого веб-сервісу. Таким чином, сервіс має багато потенційних клієнтів відразу з моменту початку роботи, а підключення до нього не вимагає істотних змін на стороні клієнта. Концепція веб-сервісів включає в себе можливість ведення реєстру веб-сервісів. Опис інтерфейсу може бути отримано з такого реєстру. Після створення і впровадження нового веб-сервісу, має сенс зареєструвати його в реєстрі. Тоді клієнти при пошуку сервісів, що реалізують вихідний інтерфейс, отримають вказівку і на новий веб-сервіс, альтернативний веб-сервіс. Існують різні сервіси, які надають якісне рішення таких задач як аутентифікація, ведення календаря, відправлення повідомлень, пошук, переклад і т. п. [7]

Використання веб-сервісу, як будівельного блоку при створенні програми передбачає що додаток може використовувати веб-сервіси як вилучені компоненти, які надають певну функціональність (рис. 2.6).

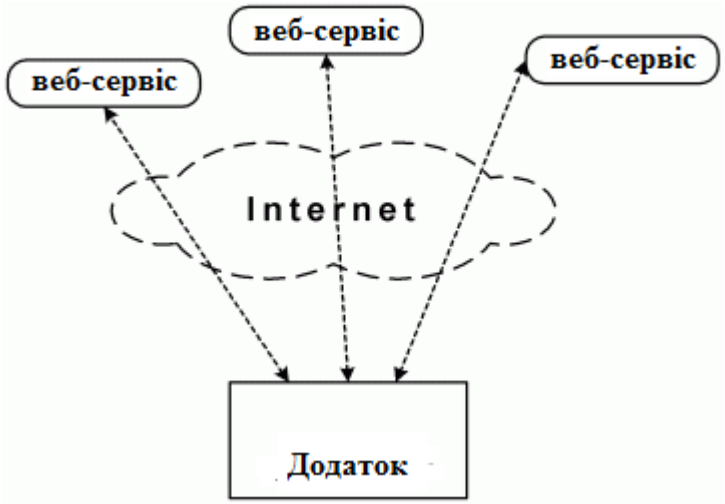


Рисунок 2.6 – Схема застосування веб-сервісу, в якості будівельного блоку

Крім зазначених сценаріїв застосування веб-сервісів, можливі інші варіанти їх. Наприклад, існують дослідження з використання веб-сервісів для

побудови розподілених обчислювальних і інформаційних систем і тимчасових і зі складною ієрархічною структурою.

2.5 Вибір СУБД

Для розробки web-сервісу служби продажу квитків, який застосовує сервіс-орієнтовану архітектуру, доцільно використовувати СУБД середнього масштабу і продуктивності. На сьогоднішній день СУБД MySQL є однією з надійних і швидких з існуючих СУБД, що найкраще підходить для систем, які функціонують у мережі Internet. СУБД MySQL написана для різних видів операційних систем, як для UNIX-сумісних операційних систем так і для ОС сімейства Windows.

Принцип роботи СУБД MySQL аналогічний принципу роботи будь-якої СУБД, що використовує SQL (Structured Query Language – мова структурованих запитів), як командної мови для створення чи видалення баз даних, таблиць, для поповнення таблиць даними, для здійснення вибірки даних [12].

MySQL, як і будь-яка інша СУБД являє собою програму-сервер, яка знаходиться в пам'яті комп'ютера і обслуговує TCP порт. А клієнтська програма, будь то CGI-додаток або програмний продукт на C, з'єднується з СУБД з цього порту і посилає йому рядки на SQL. Той у свою чергу їх інтерпретує, виконуючи необхідні дії, і відсилає результати запиту назад клієнту. Таким способом відбувається спілкування сервера баз даних з клієнтськими програмами [8].

Для запуску MySQL-сервера необхідно виконати файл mysqld.exe. Сервер запускається як без віконний фоновий процес. При цьому він залишається в пам'яті і обробляє запити від клієнтських додатків. Для зупинки сервера слід виконати команду mysqladmin – u root shutdown. Якщо сервер не був зупинений коректно, то при наступному запуску у файлі

mysql.err буде додано запис про збої. Коректна зупинка сервера необхідна для збереження всіх даних, які знаходяться в кешах MySQL [9].

MySQL має розвинену систему доступу до баз даних. Користувачеві бази даних може бути наданий доступ до всієї бази даних, окремих таблиць і окремих стовпців таблиць. Є розмежування на дії, які може виробляти користувач із записами. Для організації такої, складною структури доступу використовується декілька таблиць в системній базі даних. На підставі значень цих таблиць налаштовується політика надання доступу.

База даних, яку сервер MySQL використовує для зберігання внутрішньою інформації про користувачів, за замовчуванням має ім'я mysql. У цій базі даних певні таблиці для зберігання інформації користувача і його облікових записів.

2.6 Вибір мови програмування

Для реалізації web-сервісу служби продажу квитків була обрана об’єктно-орієнтована мова програмування PHP. Такий вибір обґрунтовується тим, що PHP має дуже зручні характеристики для реалізації поставленої задачі. Тобто для реалізації сервіс-орієнтованої системи на базі SOAP в PHP існує DLL-бібліотека, яка містить у собі важливі класи та методи, які допомагають у розробці.

У мові PHP реалізований механізм виділення ресурсів і забезпечена поліпшена підтримка об'єктно-орієнтованого програмування, засоби управління сеансом і механізм підрахунку посилань, що запобігає виділення зайвої пам'яті. Ще одна перевага полягає в тому, що вихідний текст сценаріїв PHP не можна переглянути в браузері, оскільки сценарій компілюється до його відправлення за запитом користувача. Немає проблем і з залежністю від браузерів, оскільки перед відправкою клієнту сценарії PHP повністю інтерпретується на стороні сервера. Основні фактори успіху PHP програмування це – доступність і простота. PHP має ряд особливостей, які

добре працюють разом, вони включають збір інформації, динамічне введення і низький рівень абстракції, що робить доступнішою дану мову. У тому числі доступністю дешевих і широко доступних хостинг-провайдерів, що є важливим чинником, який сприяє поширенню відкритого вихідного коду PHP [8].

PHP це особлива мова, створена спеціально, щоб використовувати її для веб-інтерфейсу. Відсутність розгортання або необхідності компілювати робить швидкі цикли зворотного зв'язку і легкі шляхи розвитку. Велике поширення і застосування означає підтримку і доступну документацію. PHP обробляється на стороні сервера і є скриптовою мовою, що вбудовується до HTML-документу, це означає, що при виборі PHP в якості мови реалізації, він дозволяє динамічно генерувати web-сторінки швидко і легко. Крім того, він доступний для більшості операційних систем і веб-серверів, а також легко співпрацює з базами даних, включаючи MySQL. Оскільки PHP не містить коду, орієнтованого на конкретний web-сервер, користувачі не обмежуються певними серверами. Apache, Microsoft IIS, Stronghold і Zeus – PHP працює на всіх перерахованих серверах. Оскільки ці сервери працюють на різних платформах, PHP в цілому є платформо-незалежною мовою й існує на таких платформах, як UNIX, Solaris, FreeBSD і Windows. PHP – мова, яка вбудована безпосередньо в html-код сторінок, які, в свою чергу будуть коректно оброблені PHP-інтерпретатором. У PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase; через стандарт відкритого інтерфейсу зв'язку з базами даних (OpenDatabaseConnectivityStandard ODBC) можна підключатися до всіх баз даних, в яких існує драйвер [8].

Ефективність є винятково важливим фактором при програмуванні для багатокористувацьких середовищ, до яких належить і Web-середовище. Дуже важлива перевага PHP полягає в його трансльованому інтерпретаторі. Такий пристрій дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість PHP-сценаріїв обробляються швидше за

аналогічні їм програми, написані на Perl. Продуктивність PHP цілком достатня для створення цілком серйозних Web-додатків. З точки зору типізації, PHP є мовою програмування з динамічною типізацією. Немає необхідності явного визначення типу змінних, хоча така можливість існує. У разі звернення до змінної, ядро PHP трактує її тип відповідно до контексту. При необхідності можливе приведення змінної певного типу за допомогою відповідних конструкцій мови. Також PHP пропонує простий і дуже функціональний інтерфейс для роботи з web-сервісами за допомогою розширення модулем SOAP, що дозволяє створювати сценарії, які обмінюються інформацією з іншими додатками за допомогою XML-пакетів, поверх існуючих протоколів, таких як HTTP [5].

Існує три стандартних класи для роботи з SOAP:

– клас SoapClient – створює SOAP-клієнта, який співпрацює з SOAP-сервісом;
– клас SoapServer – створює SOAP-сервер, для роботи клієнта;
– клас SoapFault – обробляє помилки та виводить про них інформацію.

3 ПРОЕКТУВАННЯ WEB-СЕРВІСУ

Для здійснення всіх етапів проектування web-сервісу служби продажу квитків, заснованої на використанні обраної сервіс- орієнтованої архітектури, по-перше необхідно визначити основні функціональні вимоги, що пред'являються до сервісу. Чітко визначити діапазон завдань служби і базу даних, складу її користувачів і областей застосування. На основі функціональних вимог здійснити проектування бази даних, яка забезпечить web-сервіс всіма необхідними даними.

3.1 Діаграми функціональних можливостей користувачів сервісу

Визначення функціональних можливостей користувачів є суттєвою складовою розробки web-сервісу служби продажу квитків. В сервісі розповсюдження квитків на різні заходи передбачено кілька категорій користувачів, кожен з яких визначає вимоги по відношенню до додатка бази даних в частині збережених у ній даних і транзакцій, що виконуються над даними. Тобто в системі визначається, які дії і над якими даними повинен виконувати той чи інший користувач.

Набір функцій тієї чи іншої категорії користувачів може відноситися тільки до даної категорії або частково збігатися з набором функцій інших категорій користувачів. Визначимо функціональні можливості користувачів web-сервісу служби продажу квитків на заходи, які наголошені установами-організаторами. Користувачів в системі можливо розділити на наступні групи:

- гість – незареєстрований користувач;
- привілейований користувач – це зареєстрований користувач з розширеними правами;
- адміністратор – безпосередній адміністратор системи;

– касир – користувач, який нараховує бонуси і проводить облік квитків.

Функціональні можливості користувача Гостя системи цілком є частиною функціональних можливостей Привілейованих користувачів системи, а також касира та адміністратора (рис. 3.1).

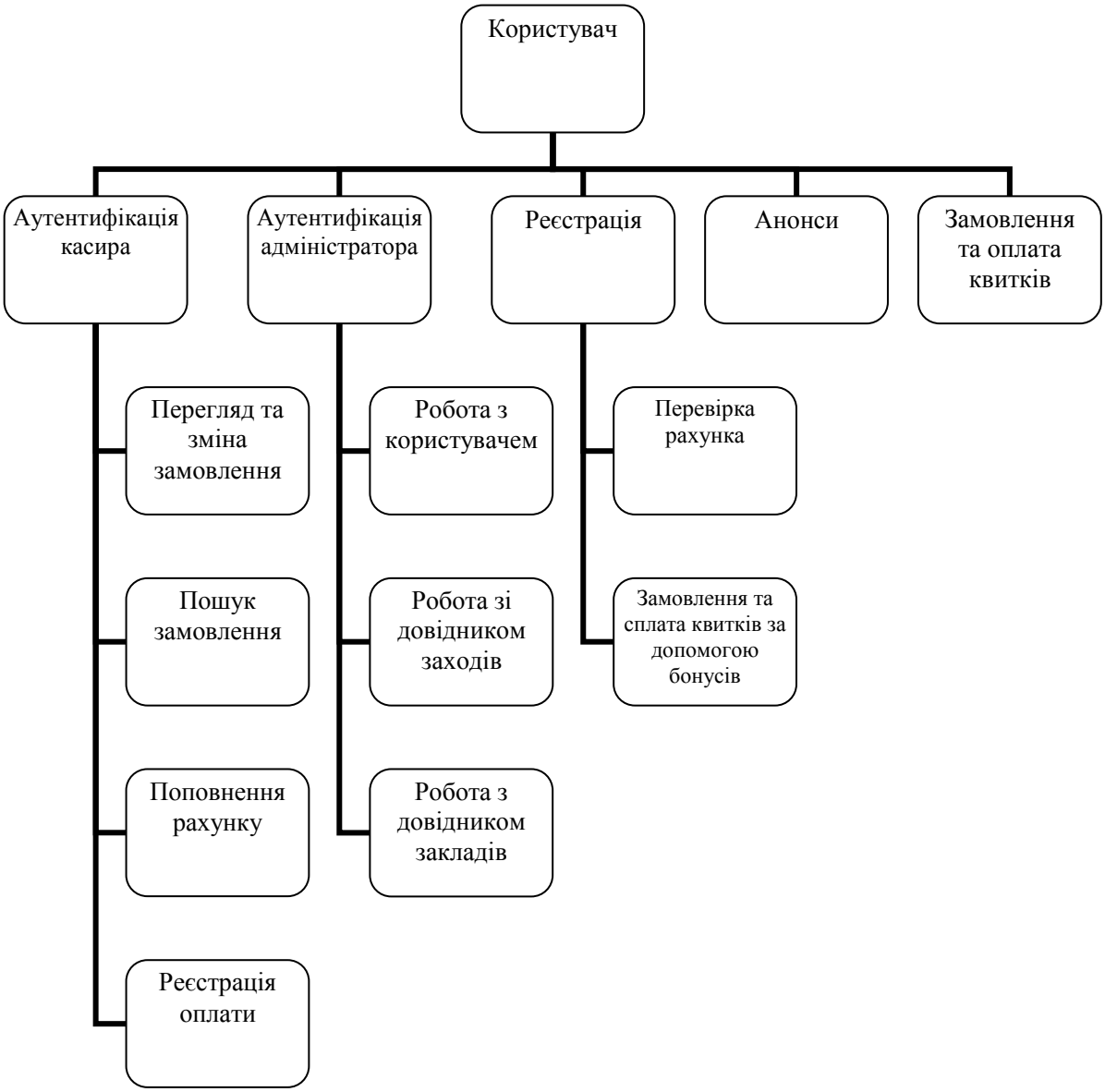


Рисунок 3.1 – Функціональні можливості всіх категорій користувачів

Далі необхідно визначити функціональні можливості всіх категорій користувачів сервісу за допомогою графічних діаграм.

Визначимо детально функціональні можливості всіх категорій користувачів сервісу. У web-сервісі служби продажу квитків передбачено клієнта Привілейований користувач – це зареєстрований користувач, якому доступні такі можливості: авторизація; перегляд анонсів; перегляд заходів; замовлення квитка; оплата квитка за допомогою бонусів або готівкою в касі. Діаграма функціональних можливостей зареєстрованого клієнта представлена на рис. 3.2.

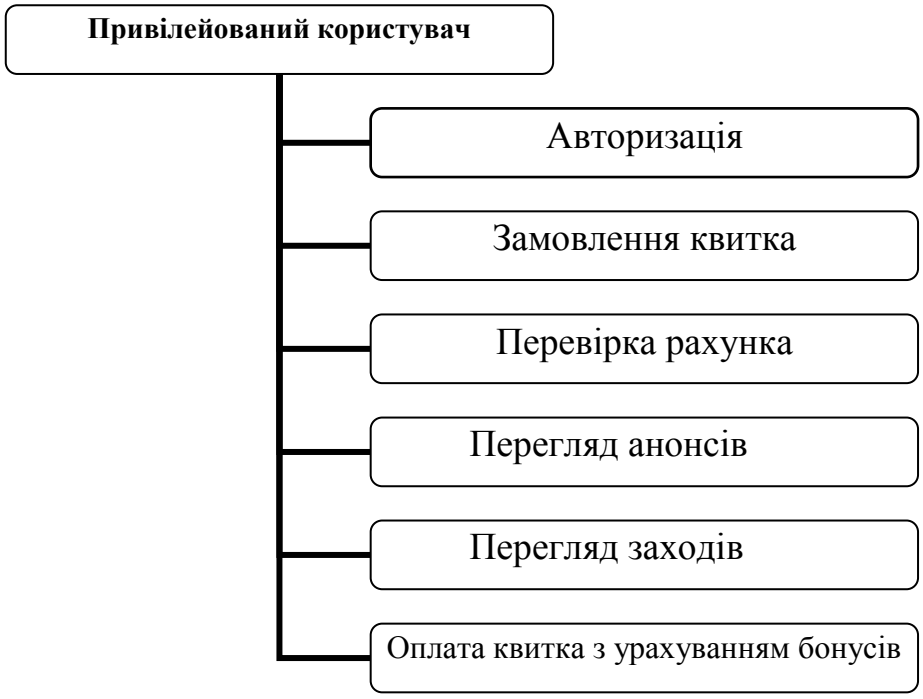


Рисунок 3.2 – Функціональні можливості зареєстрованого користувача

Клієнт Гість – це звичайний користувач глобальної мережі Internet, який завантажив в браузері сторінку web-сервісу служби продажу квитків. Йому доступні наступні функції: реєстрація – можливість зареєструватися, отримати свій логін і пароль; перегляд анонсів; перегляд заходів; замовлення квитка; оплата готівкою в касі. Діаграма функціональних можливостей користувача Гість представлена на рис. 3.3.

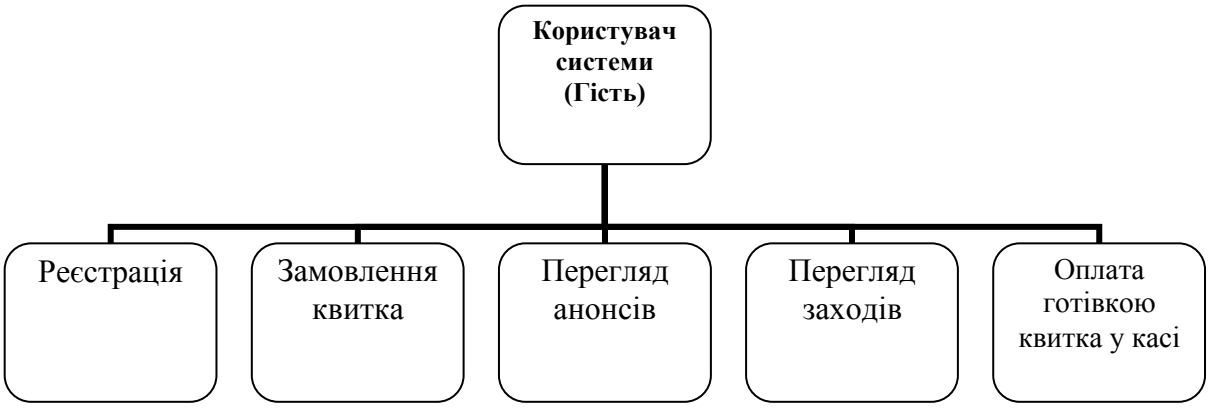


Рисунок 3.3 – Функціональні можливості Користувача Гість

Для здійснення управління сервісом передбачено користувача Адміністратора. Безпосередньо адміністратор системи має таки права: аутентифікація та авторизація; додавання, зміна, видалення заходів, закладів, типів заходів, користувачів (касирів та адміністраторів). Діаграма функціональних можливостей користувача Адміністратора представлена на рис. 3.4.

Рисунок 3.4 – Функціональні можливості Адміністратора

В web-сервісі служби продажу квитків, які проводяться установою-організатором заходів передбачено користувача Касир. Касир, який веде

облік замовлень і продаж квитків та здійснює керування бонусними рахунками Привілейованих (зареєстрованих користувачів). Безпосередньо користувач Касир сервісу має таки права: аутентифікація та авторизація; перегляд замовлень та їх зміна; пошук замовлення; поповнення рахунку; реєстрація оплати.

Діаграма функціональних можливостей користувача Касира web-сервісу служби продажу квитків представлена на рисунку 3.5.

Рисунок 3.5 – Функціональні можливості користувача Касира

За допомогою діаграм здійснено визначення функціональних можливостей всіх категорій користувачів, що дозволяє перейти до проектування бази даних web-сервісу служби продажу квитків.

3.2 Моделювання бізнес-процесів системи

Бізнес-процес – це сукупність взаємопов'язаних заходів або задач, спрямованих на створення певного продукту або послуги для споживачів. Існують три види бізнес-процесів [1]:

– керуючі – бізнес-процеси, які управляють функціонуванням системи. Прикладом керуючого процесу може служити корпоративне управління та стратегічний менеджмент;
– операційні – бізнес-процеси, які складають основний бізнес компанії і створюють основний потік доходів. Прикладами операційних бізнес-процесів є постачання, виробництво, маркетинг, продаж;
– підтримуючі – бізнес-процеси, які обслуговують основний бізнес. Наприклад, бухгалтерський облік, підбір персоналу, технічна підтримка.

У обраній предметній області підтримуються операційні бізнес-процеси, спрямовані на збільшення доходу підприємствами чи організаціями, що здійснюють анонсування проведення та організацію культурно-масових заходів. Розрізняють також основні і допоміжні процеси. Основні процеси – це ті, які додають якість, допоміжні процеси формують інфраструктуру організації. Основним таким процесом є реалізація квитків на заходи, а допоміжними бізнес-процесами є ведення бази даних всіх заходів, анонсування їх в просторі WWW мережі Інтернет та попереднє замовлення квитків клієнтами.

Інформаційна система замовлення квитків повинна підтримувати такі бізнес-функції:

– анонси майбутніх заходів;
– резервування квитків безпосередньо користувачем;
– покупка квитка в касі будь-якої установи, з яким співпрацює організатор заходів на які поширюються квитки;
– пошук потрібної інформації;
– розділення прав доступу до інформації між адміністратором, касиром, користувачем і зареєстрованим користувачем;
– для адміністратора: додавання, видалення, редагування даних предметної області;

– для касира: пошук замовлень, їх редагування, поповнення клієнтських рахунків, реєстрація оплати замовлення;
– для зареєстрованих користувачів: передоплата квитків шляхом ведення особистого рахунку користувача з урахуванням можливості поповнення рахунку та нарахування бонусів.

У кожної категорії користувачів повинен бути свій власний інтерфейс, за допомогою якого здійснюється доступ до доступних їм функцій.

Для формального опису підприємства-організатора культурно-масових заходів і його бізнес-процесів необхідно обстежити діяльність цього підприємства і зібрати достатню інформацію. Після вербального опису системи переходять до більш суворого (формального) опису – моделюванню (побудови моделей) бізнес-процесів.

З розвитком інформаційних технологій та розширенням області їх застосування зв'язок інформаційних технологій з бізнес-процесами став ще тісніше, роль моделювання бізнес-процесів істотно зросла. Для моделювання бізнес-процесів застосовується велика кількість різних методологій або, іншими словами, мов опису бізнес-діяльності підприємства чи організації. Наприклад, це технології SADT – методологія структурного аналізу і проектування (Structured Analysis and Design Technique), IDEF0 – методологія функціонального моделювання, DFD – методологія моделювання потоків даних, IDEF3 – методологія моделювання потоків робіт, IDEF1X – методологія опису даних, IDEF4 – об'єктно-орієнтована методологія, яка відображає взаємодію об'єктів, UML – (Unified Modeling Language) мова візуального моделювання, заснована на об'єктно-орієнтованому підході [9].

Для моделювання бізнес-процесів обраної предметної області застосована методологія ARIS. Абревіатура ARIS розшифровується як «Архітектура інтегрованих інформаційних систем». Поняття «архітектура» в області інформаційних технологій застосовується для опису: типів інформаційних систем, їх функціональних властивостей, взаємовідносин між окремими частинами інформаційної системи.

Методологія ARIS описує бізнес-процес у вигляді потоку послідовно виконуваних робіт. Методологія ARIS передбачає певний підхід до формалізації інформації про діяльність організації і представлення її у вигляді графічних моделей, зручному для розуміння та аналізу. Моделі, які створюються за цією методологією, відображають існуючу ситуацію з тим або іншим ступенем наближеності. Ступінь деталізації опису залежить від цілей проекту, в рамках якого проводиться моделювання. Послідовність функцій в рамках бізнес-процесів відображається у вигляді ланцюжка процесів, де для кожної функції можуть бути визначені початкова і кінцева подія. Події не тільки перемикають функції (передають управління від однієї функції до іншої), але і є їх результатом [10].

Під подією розуміється той факт, що інформаційний об'єкт отримує пов'язаний з бізнес-процесом статус, який управляє або впливає на подальше виконання бізнес-процесу. Події перемикають функції, тобто передають управління від однієї функції до іншої. Вони можуть бути також результатом виконання функцій. На відміну від функцій, які мають деяку тривалість, події відбуваються моментально.

Події перемикають функції і можуть бути результатом виконання функції. Упорядкування комбінації подій і функцій в послідовність дозволяє створити подієві діаграми процесів EPCs (Event – Driven Process Chain – ланцюжок процесу, керована подіями). За допомогою цих діаграм процедури бізнес-процесу представляються як логічні послідовності подій.

Функціональне представлення діяльності системи представимо у вигляді eEPC, як діаграми типу Стовпець, діаграми функцій і моделі типу eEPC. Моделі типу eEPC служать для докладного опису діяльності підприємства [10]. У цих моделях відображається, по-перше, послідовність виконання функцій бізнес-процесів, де для кожної функції визначені початкові і кінцеві події, по-друге, логіка виконання бізнес-процесів, по-третє, результати виконання функцій, а також виконавці бізнес-функцій, носії інформації та прикладні системи, що підтримують дані бізнес-функції.

На рисунку 3.6 представлена eEPC-діаграма типу Стовпець. У ній наведені основні бізнес-процеси проектованої інформаційної системи (Занесення інформації про заходи, Замовлення квитка та Оплата квитка), Користувачі, які виконують дані бізнес-процеси, і початкова та кінцева подія (Анонсування заходів та оплачений Квиток), керуючі даними процесами.

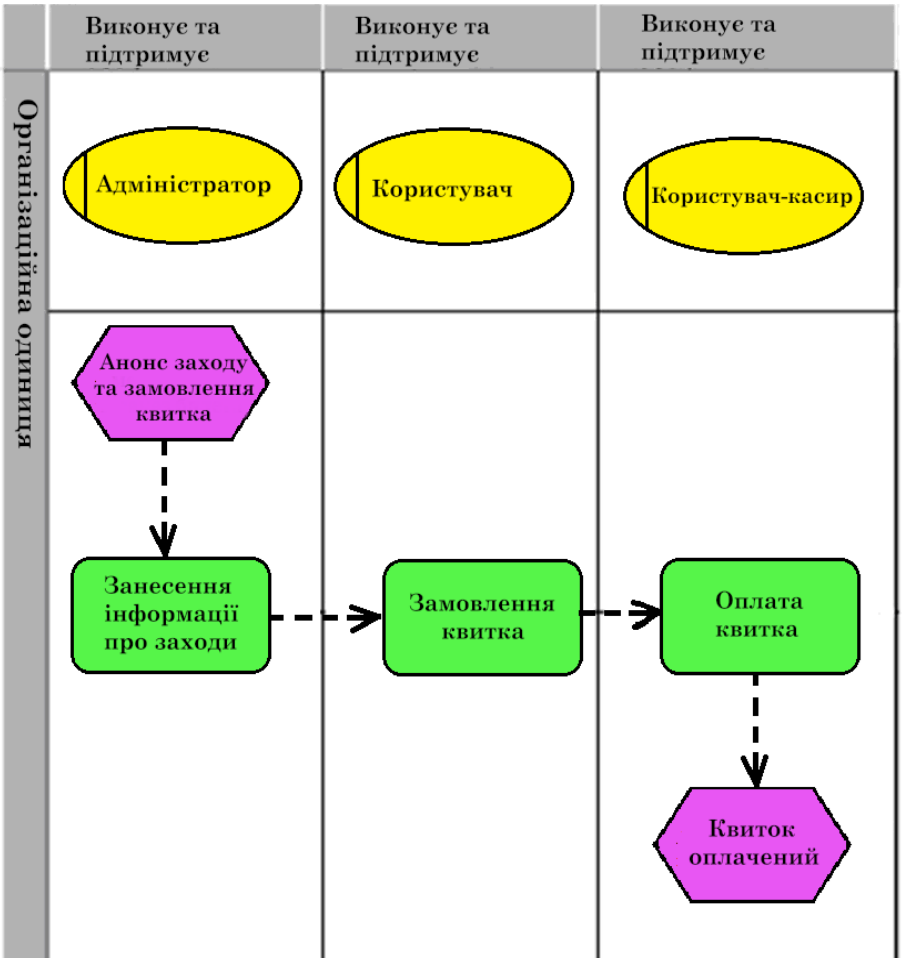


Рисунок 3.6 – eEPC-діаграма основних бізнес-процесів web-сервісу

Далі (рис.3.7 – 3.9) наведені діаграми функцій для основних бізнес-процесів web-сервісу служби продажу квитків, а саме занесення даних про заходи та замовлення квитків.

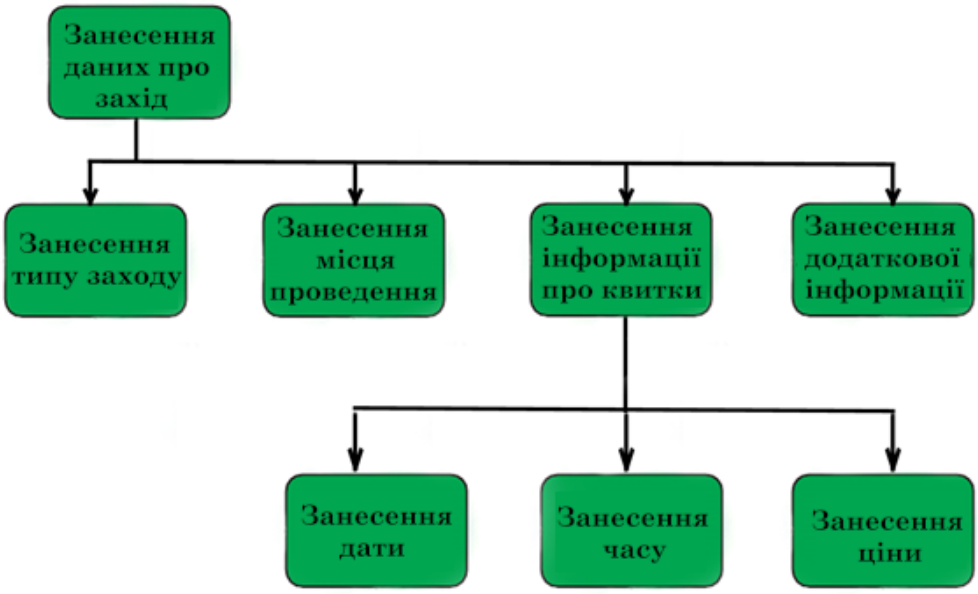


Рисунок 3.7 – Діаграма функцій для процесу «Занесення даних про заходи»

Для процесу «Занесення даних по заходи» можна визначити наступні функції: занесення типу заходу, занесення місця проведення, занесення інформації про квитки, занесення додаткової інформації. Також функція занесення даних про квитки містить декілька полів – дата, час, ціна.

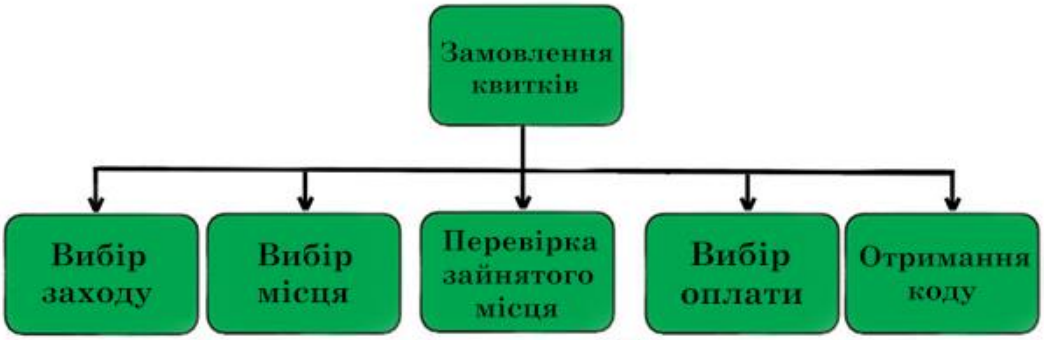


Рисунок 3.8 – Діаграма функцій для процесу «Замовлення квитків»

Для процесу «Замовлення квитків» можна визначити наступні функції: вибір заходу, місця, перевірка зайнятого місця, вибір оплати, отримання коду. На рисунку 3.9 визначена діаграма функцій для процесу «Оплата квитків».

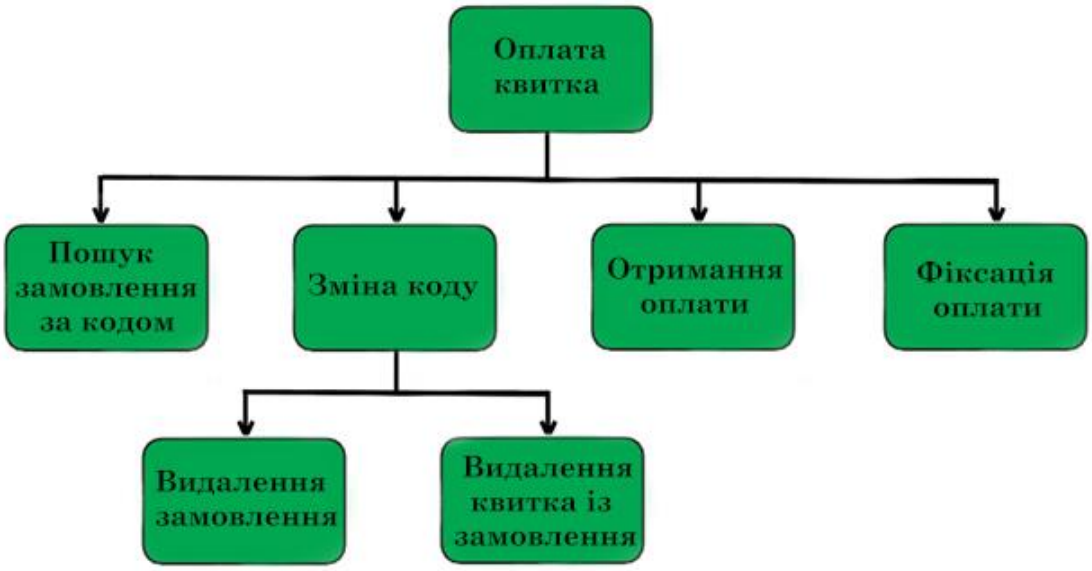


Рисунок 3.9 – Діаграма функцій для процесу «Оплата квитка»

Для процесу «Оплата квитків» можна визначити наступні функції: пошук замовлення по коду, зміна, коду, отримання оплати, фіксування оплати. Також функція зміна коду містить декілька полів – видалення замовлення та видалення квитка із замовлення.

3.3 eEPC-моделі діяльностей web-сервісу служби

Упорядкування комбінації подій і функцій в послідовність дозволяє створити подієві діаграми процесів EPCs (Event – Driven Process Chain – ланцюжок процесу, керована подіями). За допомогою цих діаграм процедури бізнес-процесу представляються як логічні послідовності подій.

Функціональне представлення діяльності web-сервісу служби продажу квитків представимо у вигляді eEPC. Моделі типу eEPC служать для докладного опису діяльності установи, що організує заходи, на які поширюються квитки. У цих моделях відображається послідовність виконання функцій бізнес-процесів, де для кожної функції визначені початкові і кінцеві події, логіка виконання бізнес-процесів, результати

виконання функцій, виконавці бізнес-функцій, носії інформації та прикладні системи, що підтримують дані бізнес-функції.

Наведемо також кілька eEPC-моделей, на різних рівнях абстракції, які описують діяльність установи, яка забезпечує організацію того чи іншого заходу та систему розповсюдження квитків, що описує діяльність цієї установи (рис. 3.10 – 3.13).

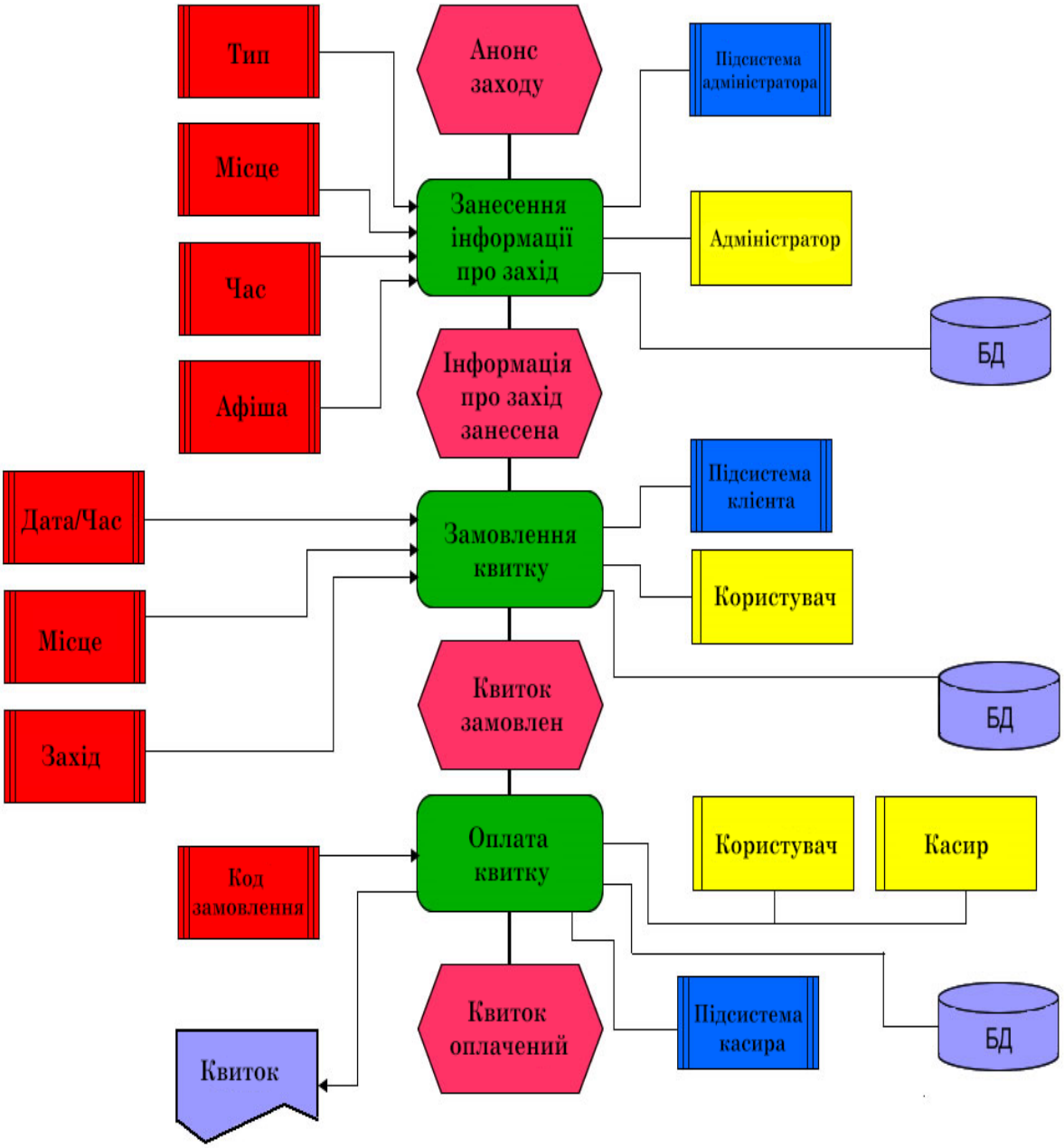


Рисунок 3.10 – eEPC-діаграма основних бізнес-процесів web-сервісу

Визначимо eEPC-модель, яка описує процес «Занесення даних про заходи», якій відповідає за авансування та інформаційну підтримку установи, що здійснює організацію заходів на які розповсюджуються квитки (рис.3.11).

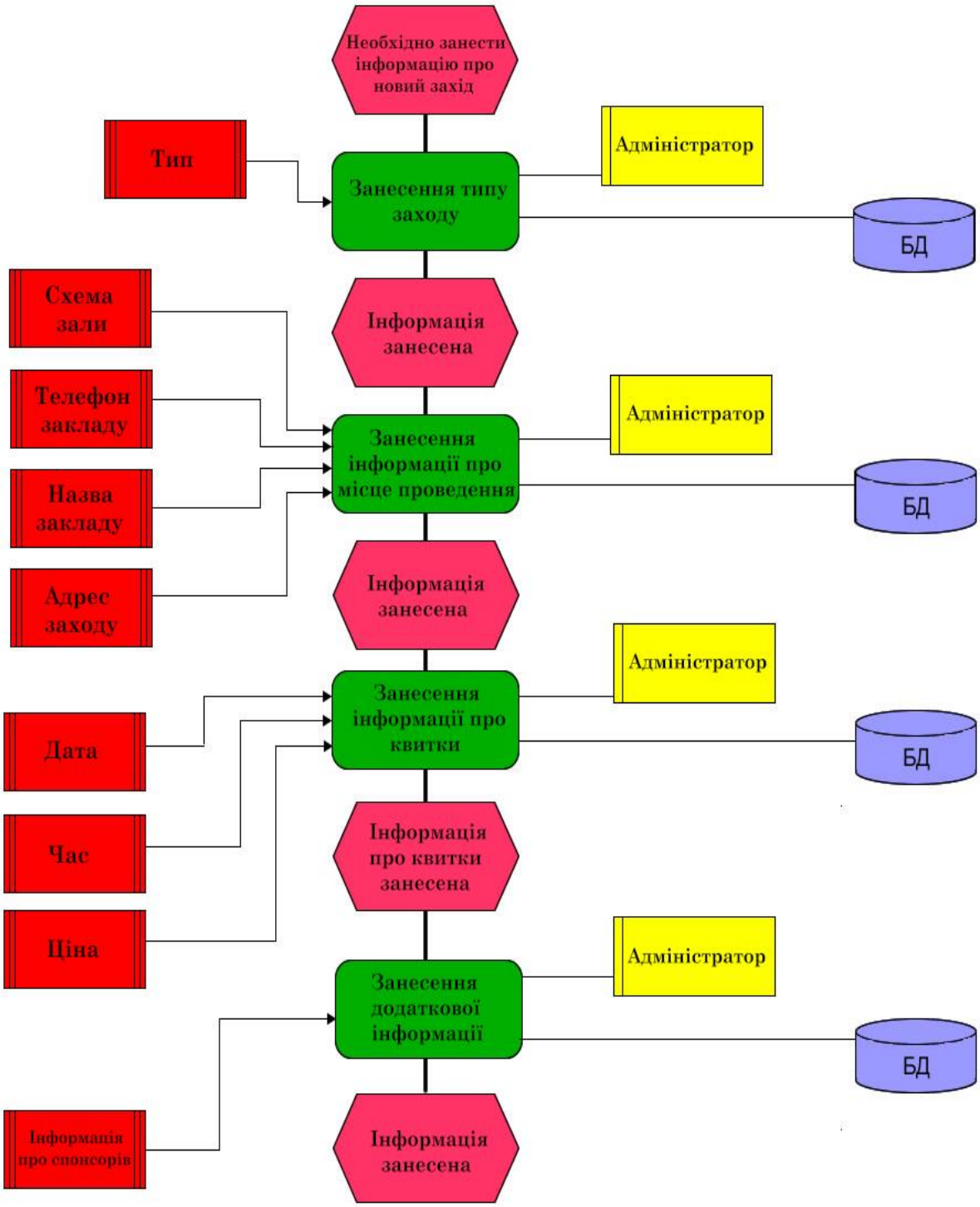


Рисунок 3.11 – eEPC-діаграма для процесу «Занесення даних про заходи»

Визначимо eEPC-модель, яка описує процес «Замовлення квитка», яка забезпечує здійснення процесу бронювання у web-сервісі (рис. 3.12).

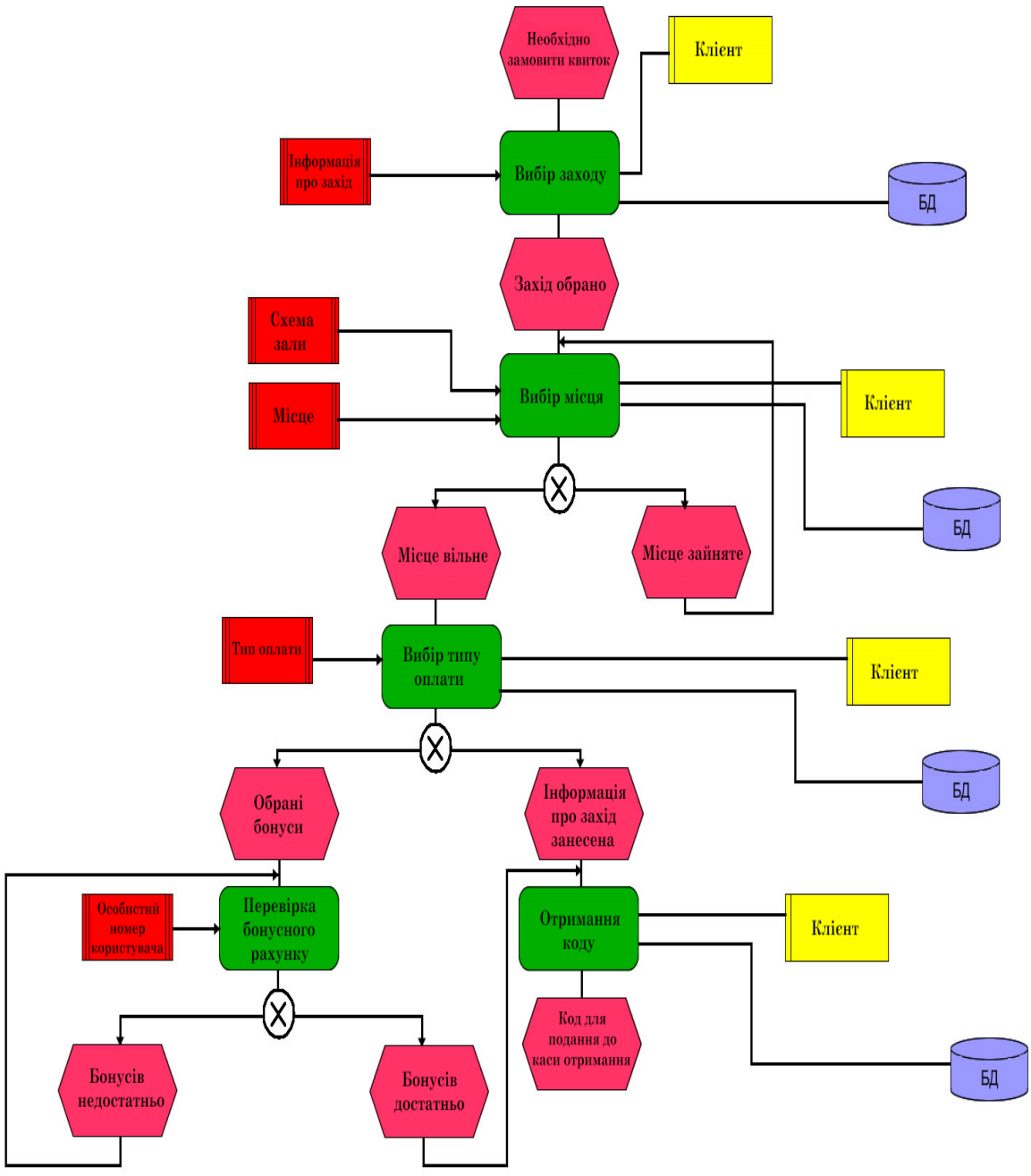


Рисунок 3.12 – eEPC-діаграма для процесу «Замовлення квитка»

Для процесу «Оплата квитка» була визначена eEPC-модель, яка описує процес обробки замовлення, нарахування бонусів та здійснення оплати Користувачем-касиром (рис. 3.13).

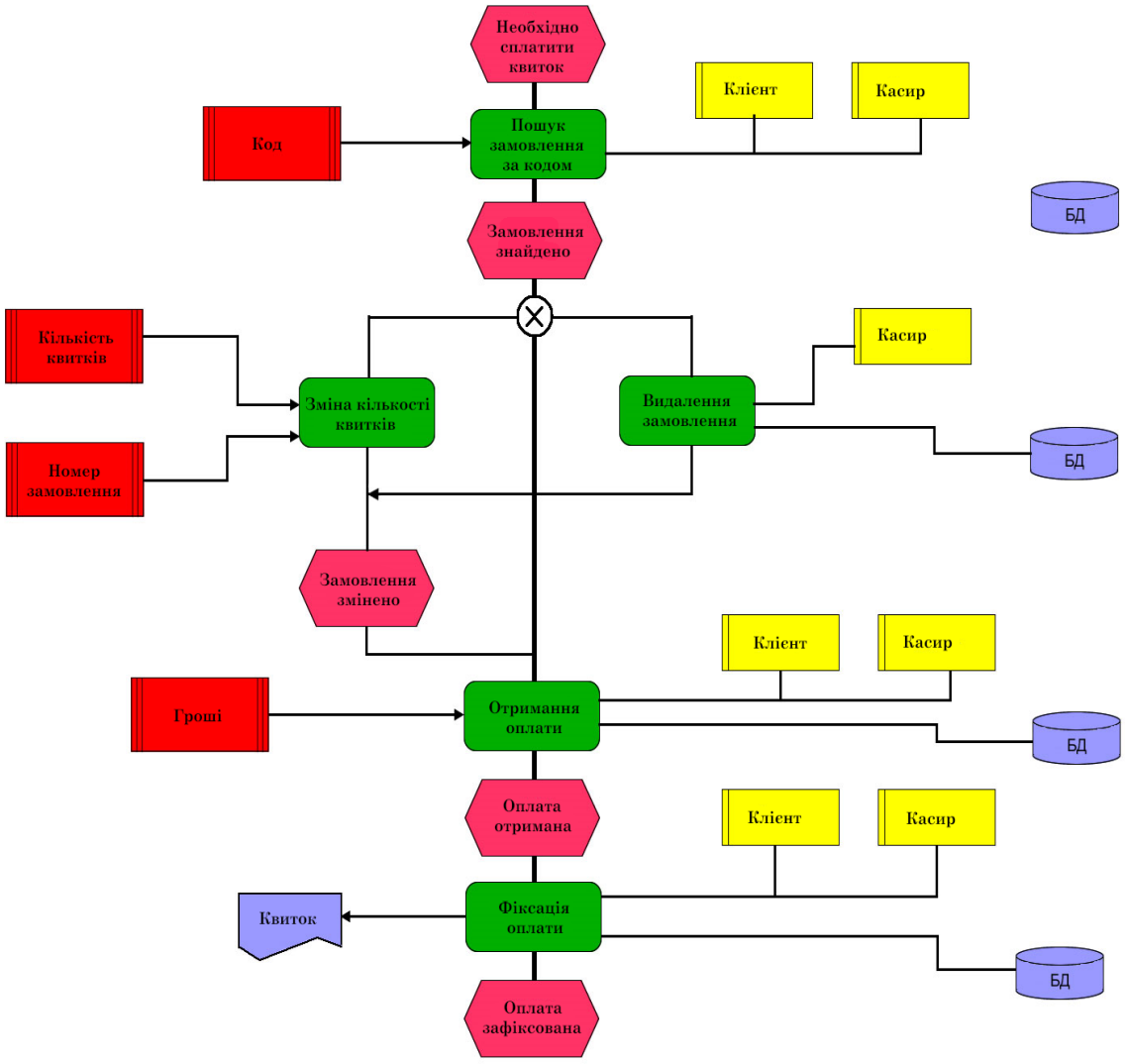
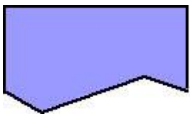
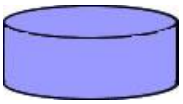


Рисунок 3.13 – eEPC-діаграма для процесу «Оплата квитка»

Моделі обраної предметної області, створювані за методологією ARIS, відображають існуючу ситуацію з тим або іншим ступенем наближеності. Ступінь деталізації опису залежить від цілей проекту, в рамках якого проводиться моделювання. В результаті побудовані моделі, які відображають функціональне представлення діяльності web-сервісу служби продажу квитків на заходи, згідно з використаною методологією ARIS, описує бізнес-процес у вигляді потоку послідовно виконуваних робіт.

Основні об'єкти методології моделювання ARIS, використовувані при моделюванні обраної предметної області, використані в даній роботі наведені в таблиці 3.1.

Таблиця 3.1 – Основні об'єкти методології моделювання ARIS

Позначення	Назва	Призначення
	Подія	Служить для опису станів системи, керуючих виконанням функцій
	Функція	Служить для опису функцій, які виконуються підрозділами або співробітниками установи.
	Прикладна система	Служить для опису прикладних систем, що використовуються в рамках виконання функцій
	Посада	Служить для опису організаційних одиниць, що виконують функції
	Документ	Служить для опису об'єктів, що відображають реальні носії інформації, наприклад, паперові документи
	Носій інформації	Служить для опису баз даних
	Кластер інформації	Служить для опису наборів даних як наборів сутностей і зв'язків між ними

В результаті виконаного моделювання підтримуються операційні бізнес-процеси, спрямовані на збільшення доходу установи, яка організує заходи. Виявлені основні процеси, якими є розповсюдження квитків на заходи, а допоміжними бізнес-процесами є ведення бази даних всіх заходів, анонсування їх в мережі Інтернет і попереднє замовлення квитків клієнтами.

3.4 Проектування бази даних системи

Виходячи з аналізу предметної області, для функціонування web-сервісу служби продажу квитків, який засновано на архітектурі веб-сервісів,

необхідно здійснити проектування бази даних (БД), в якій повинні зберігатися наступні відомості:

- про приміщення установ, де можуть проводитися концерти, виставки, конференції тощо;
- про розташування місць в концертних залах;
- про поточні та майбутні заходи;
- про ціни на квитки;
- про наявність вільних місць;
- про користувачів-власників карток і сум грошей на рахунках;
- про замовлення квитків на проведені організацією заходи.

Основними об'єктами в розглянутій предметній області є: Заклад (приміщення); Захід; Користувачі; Замовлення.

В базі даних web-сервісу служби продажу квитків необхідно зберігати також і наступну інформацію:

- про Заклади: назва, адреса, телефон, адреса web-сайту, тип (нічний клуб, театр, виставковий зал, конференц-зал і т.п.).
- про Заходи: назва, дата, час проведення, опис, інформація про спонсорів, місце проведення, вартість квитків.
- про Користувачів: ПІБ, телефон, адреса електронної пошти, кількість грошей (бонусів) на рахунку, пароль для входу в систему.
- про Місця: розташування місць описується зоною залу для глядачів у відповідному закладі, номером ряду і номером місця.
- про Замовлення: замовник, дата замовлення, квитки на захід, пароль(шифр необхідний, коли користувач замовляє квиток за допомогою Internet для пред'явлення його в касі і подальшого отримання замовлених квитків), тип оплати (готівкою або за допомогою бонусів на особовому рахунку);
- про Привілейованих користувачів: логін і пароль для входу в систему, тип користувача (касир, адміністратор).

Оскільки в базі даних повинна бути виключена надмірність інформації, а також аномалії оновлення, додавання і видалення, необхідно зробити декомпозицію вихідного набору таблиць і привести базу даних до нормальної форми. Після проведення процедури нормалізації БД приведена до 3-ї нормальної форми, тобто всі не ключові атрибути не будуть транзитивно залежати від ключових атрибутів сутностей. Для виключення аномалій оновлення, додавання і видалення, а також надмірності інформації необхідно провести процедуру нормалізації, а саме виконати [11]:

- виділити Типи заходів в окрему сутність, а Заходи визначити посилання на сутність Типи заходів шляхом додавання зовнішнього ключа;
- виділити Типи користувачів в окрему сутність, а по суті Користувачі визначити посилання на сутність Типи користувачів шляхом додавання зовнішнього ключа;
- виділити Заходи в окрему сутність, а по суті Ціни визначити посилання на сутність Заходи шляхом додавання зовнішнього ключа;
- виділити Зони в окрему сутність, а по суті Місця визначити посилання на сутність Зони шляхом додавання зовнішнього ключа;
- виділити Тип заходу в окрему сутність, а по суті Заходи визначити посилання на сутність Тип заходу шляхом додавання зовнішнього ключа;
- виділити Місця в окрему сутність, а по суті Замовлення визначити посилання на сутність Місця шляхом додавання зовнішнього ключа;

Між сутностями Заходи та Замовлення існує зв'язок типу «багато-до-багатьох», тому одне замовлення може враховувати бронювання кількох квитків на кілька заходів і, з іншого боку, один захід може враховуватися в декількох замовленнях.

Для реалізації зв'язку типу «багато-до-багатьох» створюємо третю сутність, помістивши в неї первинні ключі сутностей Заходи та Замовлення.

Виходячи з вимог до сервісу, а також з вимог до збереженої інформації, після проведення процедури нормалізації було отримано ряд таблиць бази даних. Сутність «КОРИСТУВАЧІ» призначена для того, щоб зберігати дані про всіх користувачів, що мають доступ до системи.

Дана сутність містить наступні атрибути (табл. 3.2): ідентифікатор користувача, ім'я користувача, прізвище користувача, по-батькові користувача, електронна адреса користувача, телефон користувача, особливий шифр користувача, кількість грошей (бонусів) на рахунку користувача.

Таблиця 3.2 – Сутність «КОРИСТУВАЧІ» (users)

№ п/п	Атрибут	Тип
1	id_user	int(2)
2	first	varchar(255)
3	last	varchar(255)
4	patro	varchar(50)
5	e_ mail	varchar(50)
6	tel	varchar(255)
7	kod	varchar(15)
8	number	varchar(50)

Сутність «ЗАКЛАД» призначена для зберігання даних про місця розташування закладів проведення заходів. Сутність містить наступні атрибути (табл.3.3): ідентифікатор закладу, назва закладу, адреса закладу, телефон закладу, адреса сайту закладу, зображення закладу.

Таблиця 3.3 – Сутність «ЗАКЛАД» (institution)

№ п/п	Атрибут	Тип
1	id	int(10)
2	name	varchar(200)
3	adres	varchar(50)
4	tel	varchar(15)
5	url	varchar(50)
6	img	varchar(200)

Сутність «ТИП ЗАХОДУ» призначена для зберігання даних про захід, який передбачається організатором. Сутність містить наступні атрибути (табл. 3.4): ідентифікатор заходу, назва заходу.

Таблиця 3.4. – Сутність «ТИП ЗАХОДУ» (type_action)

№ п/п	Атрибут	Тип
1	id	int(10)
2	name	varchar(200)

Сутність «ЗОНА» призначена для зберігання даних по культурну подію, яка передбачається підприємством-організатором. Сутність містить наступні атрибути (табл. 3.5): ідентифікатор зони, назва зони.

Таблиця 3.5 – Сутність «ЗОНА» (zone)

№ п/п	Атрибут	Тип
1	id	int(10)
2	id_inst	int(10)
3	name	varchar(30)

Наступна сутність «МІСЦЯ» призначена для зберігання даних про місця (їх розташування у залі, щоб отримати вичерпну інформацію для придбання необхідного квитка) у закладах. Сутність містить наступні атрибути (табл. 3.6): ідентифікатор місця, ідентифікатор зони, ряд, де розташоване місце, номер місця.

Таблиця 3.6 – Сутність «МІСЦЯ» (place)

№ п/п	Атрибут	Тип
1	id	int(11)
2	id_zone	int(11)
3	n_rad	int(5)
4	n_place	int(5)

Сутність «ЗАМОВЛЕННЯ» призначена для зберігання даних про всі необхідні данні для проведення замовлення квитків на обрані клієнтом

заходи. Сутність містить наступні атрибути (табл. 3.7): ідентифікатор замовлення, користувача, дата замовлення, генерований автоматично шифр замовлення, який пред'являється в касі при оплаті замовлення, тип оплати (з рахунку на картці у вигляді бонусів або готівкою в касі закладу).

Таблиця 3.7 – Сутність «ЗАМОВЛЕННЯ» (order)

№ п/п	Атрибут	Тип
1	id	int(11)
2	id_ user	int(11)
3	id_ data	int(11)
4	kodes	varchar(255)
5	card_or_cash	int(2)

Сутність «ЗАХОДИ» призначена для зберігання даних про всі заходи, передбачені сервісом (табл. 3.8).

Таблиця 3.8 – Сутність «ЗАХОДИ» (action)

№ п/п	Атрибут	Тип
1	id	int(10)
2	act_date	varchar(15)
3	name	text
4	info	text
5	id_inst	int(11)
6	type_action	int(11)
7	sponsors	text
8	img	varchar (255)

Сутність містить наступні атрибути: ідентифікатор заходу, дату проведення заходу, назва заходу, інформація про захід, тип заходу, спонсори, зображення для заходу.

Сутність «ЦІНИ» призначена для зберігання даних про ціни на квитки, згідно з обраною зоною. Сутність містить наступні атрибути: ідентифікатор ціни, ідентифікатор заходи, ідентифікатор зони, ціна (табл. 3.9).

Таблиця 3.9 – Сутність «ЦІНИ» (cost)

№ п/п	Атрибут	Тип
1	id	int(11)
2	id _ action	int(11)
3	id_zone	int(11)
4	cost	varchar(255)

Сутність «ПРИВІЛЕЙОВАНІ КОРИСТУВАЧІ» призначена для того, щоб зберігати дані про всіх зареєстрованих користувачів, які мають доступ до управління інформацією у сервісі. Ця сутність містить наступні атрибути (табл.3.10): ідентифікатор користувача, його логін, пароль, ім’я користувача, тип користувача.

Таблиця 3.10 – Сутність «ПРИВІЛЕЙОВАНІ КОРИСТУВАЧІ» (admins)

№ п/п	Атрибут	Тип
1	id	int(11)
2	login	varchar(255)
3	pass	varchar(255)
4	name	varchar(50)
5	type _ adm	int(11)

Сутність «ТИПИ ПРИВІЛЕЙОВАНИХ КОРИСТУВАЧІВ» містить наступні атрибути (табл. 3.11): ідентифікатор типу, назва типу. Наявнысть у базы даних сутності «ТИПИ ПРИВІЛЕЙОВАНИХ КОРИСТУВАЧІВ» дозволяє уникнути збитковості даних та розподілити доступ до інформації у системі.

Таблиця 3.11 – Сутність «ТИПИ ПРИВІЛЕЙОВАНИХ КОРИСТУВАЧІВ»

№ п/п	Атрибут	Тип
1	id	int(11)
2	name	varchar(255)

Після того, як всі сутності та всі атрибути кожної з сутностей для створення бази даних web-сервісу служби продажу квитків були визначені та описані, була розроблена діаграма «Сутність-Зв'язок» бази даних (рис. 4.6).

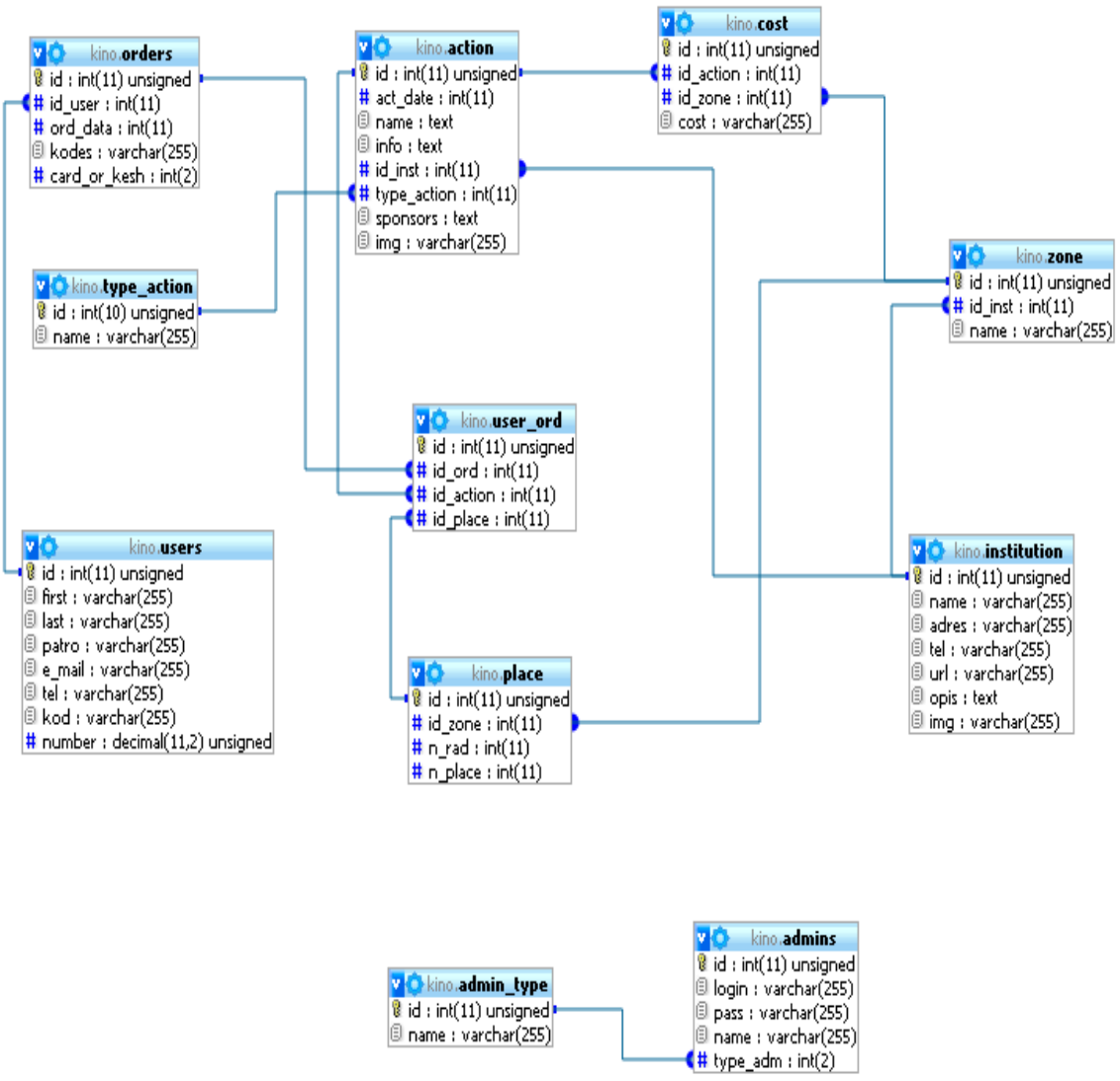


Рисунок 4.6 – Діаграма «Сутність – Зв'язок» бази даних сервісу

Далі описані зв'язки між таблицями бази даних web-сервісу служби продажу квитків.

Між таблицями Користувачі та Замовлення існує зв'язок типу один-до-багатьох: один і той же користувач може кілька разів робити замовлення.

Між таблицями Заходи та Ціни існує зв'язок типу один-до-багатьох: на один і той же захід можуть бути різні ціни в залежності від розташування місць у залі.

Між таблицями Зони і Ціни існує зв'язок типу один-до-багатьох: ціна квитків в одну і ту ж зону може варіюватися залежно від заходу.

Між таблицями Зони і Місця існує зв'язок типу один-до-багатьох: в одній зоні може бути багато місць.

Між таблицями Типи заходів та Заходи існує зв'язок типу один-до-багатьох: одному типу може відповідати кілька заходів.

Між таблицями Заклади та Заходи існує зв'язок типу один-до-багатьох: в одному і тому ж закладі можуть проходити кілька заходів.

Між таблицями Заклади і Зони існує зв'язок типу один-до-багатьох: в одному і тому ж закладі може бути кілька зон.

Між таблицями Привілейовані користувачі і Типи користувачів існує зв'язок типу один-до-багатьох: одного й того ж типу може відповідати декілька користувачів.

Таким чином, для здійснення програмної реалізації web-сервісу служби продажу квитків було визначено: функціональні можливості, які потрібно реалізувати для кожної категорії користувачів; розроблена БД, що буде містити всю необхідну інформацію для функціонування сервісу.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-СЕРВІСУ

4.1 Визначення взаємодії веб-сервісів в інформаційній системі

З урахуванням бізнес-логіки кожного з додатків користувачів web-сервісу служби продажу квитків було реалізовано три незалежні веб-сервіси, які функціонують у єдиній службі замовлення квитків та повністю реалізують необхідні функції. Веб-сервіси, які можуть бути частками розподіленої інформаційної системи, реалізують функціональні можливості користувачів системи (Користувач-Гість, Привілейований користувач, Адміністратор, Касир), що дозволить здійснювати діяльність підприємства-організатора культурно-масових заходів в мережі Інтернет та здійснювати анонсування заходів, замовлення та продаж квитків. Взаємодія PHP-скриптів для реалізації визначених веб-сервісів представлена на рисунку 4.1.

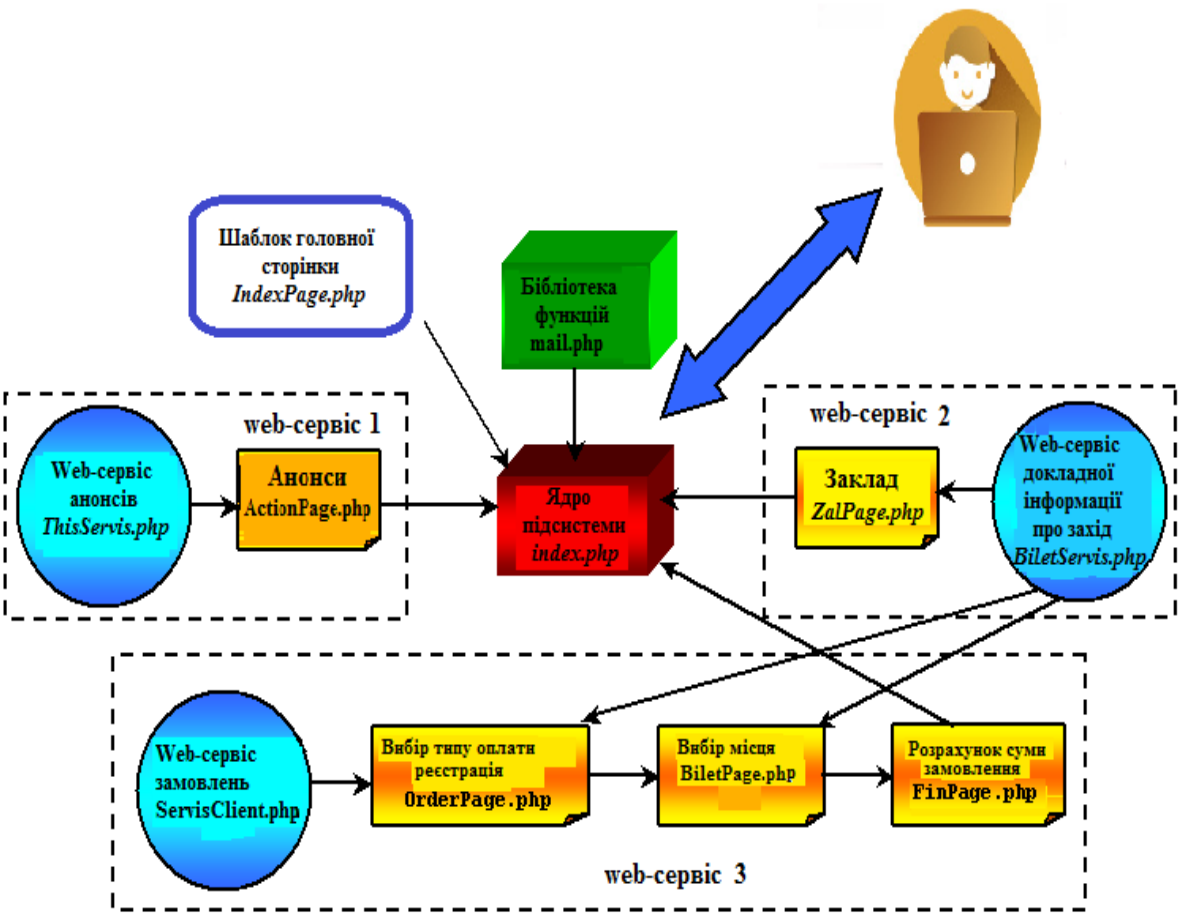


Рисунок 4.1 – Схема взаємодії розроблених веб-сервісів

Для реалізації функціональних можливостей сервісу використані програмні засоби реалізації, такі як мова програмування PHP, СУБД MySQL, бібліотека PHP для реалізації веб-сервісів php_soap.

4.2 Створення SOAP-клієнта та SOAP-серверу

Simple Object Access Protocol (SOAP) є розширення для PHP, яке дозволяє створювати самі web-служби і засобами цього протоколу створювати написання до них клієнтських додатків. Для розробки web-сервісів SOAP в PHP існує спеціальний модуль php_soap, що містить необхідні функції. Поведінка всіх цих функцій залежить від установок в конфігураційному файлі інтерпретатора PHP php.ini [12]. Для функціонування SOAP необхідно підключити модуль php_soap.dll в налаштуваннях php. У цьому файлі є три стандартних класи для роботи з SOAP: клас SoapClient, клас SoapServer, клас SoapFault. Підтримка RPC (Remote Procedure Calls) на початку була однією з незначних можливостей протоколу SOAP, але зараз вона перетворилася в одну з найбільш часто використовуваних можливостей. Клас SoapClient (рис. 4.2) створює SOAP-клієнта для web-сервісу.

```
class SoapClient {
        __construct  ( mixed wsdl [, array options] )
}
<?php
        $link ='http://www.webservicex.net/CurrencyConvertor.asmx?wsdl';
        $client = new SoapClient ($link);
        $usd  = $client->ConversionRate(array('FromCurrency' => 'USD',

        $euro = $client->ConversionRate(array('FromCurrency' => 'EUR',
        echo 'USD: '.$usd->ConversionRateResult .
                        '<br>EUR: '.$euro->ConversionRateResult;
?>
```

Рисунок 4.2 – Клас SoapClient

Даний клас має такі основні функції: створює новий об’єкт класу SoapClient; в параметрі wsdl вказується URI WSDL-документу web-сервісу; всі методи web-сервісу стають методами створеного об’єкту. Для отримання списку SOAP-функцій, необхідних для функціонування інформаційної системи замовлення квитків, застосовано наступний код (рис. 4.3), за допомогою якого повертається список SOAP-функцій, які визначені розробником та підтримуються web-сервісом.

```
class SoapClient {
        array  __getFunctions  ( void  )
}
<?php
        $client = new SoapClient ('some.wsdl');
        foreach ($client->_|getFunctions() as $type){
                echo $type . '<br>';
        }
?>
```

Рисунок 4.3 – Функції SOAP

Для отримання останнього SOAP-запиту необхідно застосувати наступний код, який повертає останній SOAP-запит (рис. 4.4). Отримання SOAP-запиту можливо при умові, що SOAP-клієнт був створений із активованою опцією trace.

```
class SoapClient {
        string __getLastRequest ( void )
}
<?php
        $client = SoapClient ("some.wsdl",array('trace' => 1));
        $result = $client->SomeFunction();
        echo "Запрос: " . $client->__getLastRequest();
?>
```

Рисунок 4.4 – Повернення останнього SOAP-запиту

Для отримання останньої SOAP-відповіді застосовують наступний код (рис. 4.5), який повертає останню SOAP-відповідь. Працює лише в тому випадку, якщо об’єкт класу був створений з активованою опцією trace.

```
class SoapClient {
        string __getLastResponse ( void )
}
<?php
        $client = SoapClient ("some.wsdl",array('trace' => 1));
        $result = $client->SomeFunction();
        echo "Ответ: " . $client->__getLastResponse();
?>
```

Рисунок 4.5 – Повернення останньої SOAP-відповіді

SOAP-сервер створює новий об’єкт класу SoapServer. У параметрі wsdl (рис. 4.6) вказується URI WSDL-документу, що використовується для створення web-сервісу.

```
class SoapServer {
        __construct ( mixed wsdl [, array options] )
}
$server = new SoapServer ("stock.wsdl");
```

Рисунок 4.6 – Створення SOAP-серверу

Для реалізації розподіленої сервіс-орієнтованої інформаційної системи замовлення квитків на культурно-масові заходи відбувається реєстрація методів web-сервісу. У параметрі functions вказується ім’я функції, що

реєструється або масив, що містить у собі імена декількох функцій, що реєструються.

Функція handle() запускає обробку web-серверу (рис. 4.7). У параметрі soap_request вказується SOAP-запит для обробки. Проте, якщо параметр soap_request опущений, то використовується значення задане за замовчуванням у змінній $HTTP_RAW_POST_DATA [12].

```
class SoapServer {
        void addFunction (mixed functions)
        void handle  ( [string soap_request] )
}
$server->addFunction("getStock");
$server->handle();
```

Рисунок 4.7 – Реалізація методу сервера та обробка запиту

Результат роботи створених веб-сервісів було протестовано та розглянуто у розподіленої інформаційній системі, що забезпечує роботу підприємства-організатора культурно-масових заходів з анонсування культурно-масових заходів та забезпечує замовлення квитків засобами мережі Інтернет.

4.3 Керівництво користувача системи

Створення web-сервісів для інформаційної системи підприємства-організатора культурних заходів дозволяє використовувати їх можливості у составі будь-якої розподіленої системі.

Розглянемо інформаційну Інтернет-систему в обраній предметній області для якої розроблені web-сервіси застосовуються як будівельні блоки. У верхній частині системи розміщено логотип та назва системи, ліворуч

розташоване меню, яке містить типи культурних заходів: концерти, вистави, семінари, тощо. В області контента розмістяться основні інформаційні блоки обраного пункту меню (рис. 4.8).

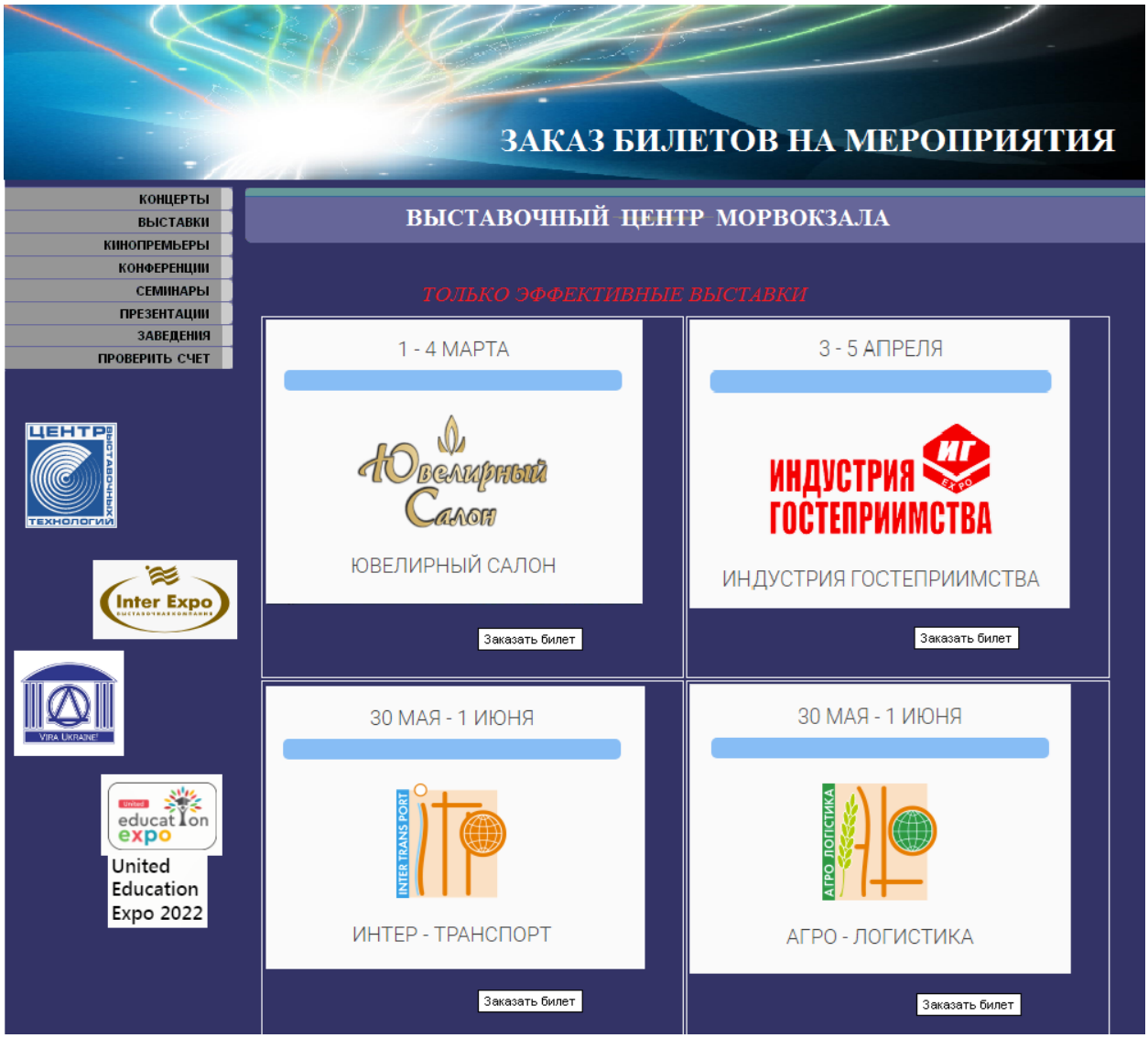


Рисунок 4.8 – Головна сторінка сервісу замовлення квитків

У клієнтській частині реалізована можливість проглядати інформацію про заклади, де буде відбуватися культурний захід. Наведена схема закладу, де позначені зони, які допомагають користувачу обирати квитки у залі для зручнішого розташування (рис. 4.9).

Рисунок 4.9 – Відображення інформації про заклад

Розроблений web-сервіс дозволяє здійснювати замовлення квитків. Процедура замовлення квитків розбита на декілька частин. Для початку необхідно вказати системі чи є ви зареєстрованим користувачем, якщо ні, то система запропонує користувачу заповнити реєстраційну форму (рис. 4.10).

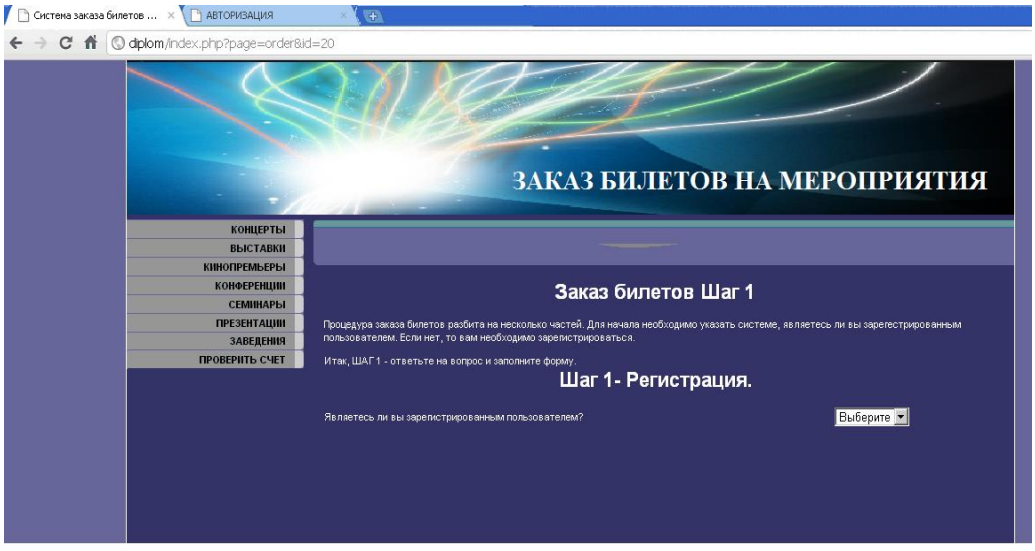


Рисунок 4.10 – Регістрація кліента

Зареєстрованому клієнту для здійснення входу в систему необхідно заповнити запропоновану системою форму, де потрібно здійснити введення персонального коду доступу (рис. 4.11).

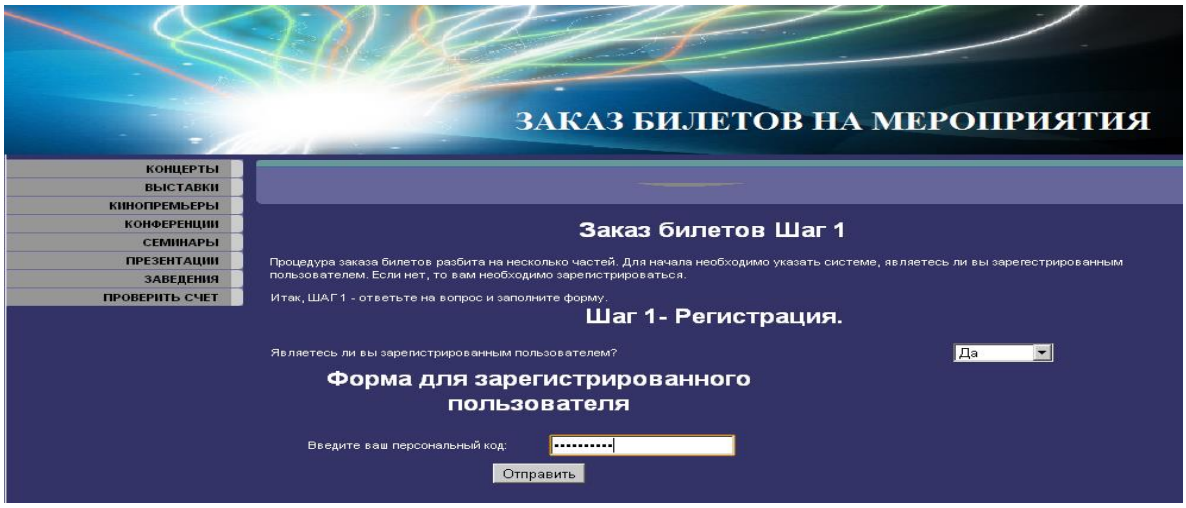


Рисунок 4.11 – Форма для ввода данних Зарегистрованого Користувача

При здійсненні процедури замовлення квитків, зареєстрований користувач може здійснити перевірку власного рахунка, де зберігається кількість отриманих користувачем бонусів. Для перевірки рахунка необхідно ввести персональний код на формі перевірки особових рахунків (рис. 4.12).

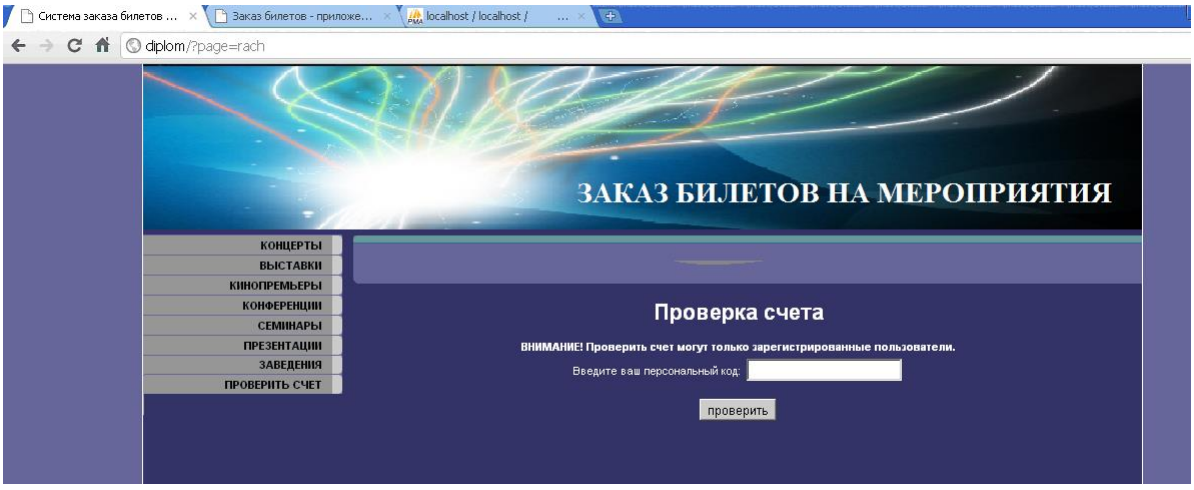


Рисунок 4.12 – Форма перевірки кількості бонусів на особовому рахунку

Після коректного введення коду персонального рахунку, система повертає повідомлення о кількості бонусів на рахунку. Для здійснення операції поповнення власного рахунку система пропонує звернутись до касира (рис. 4.13).

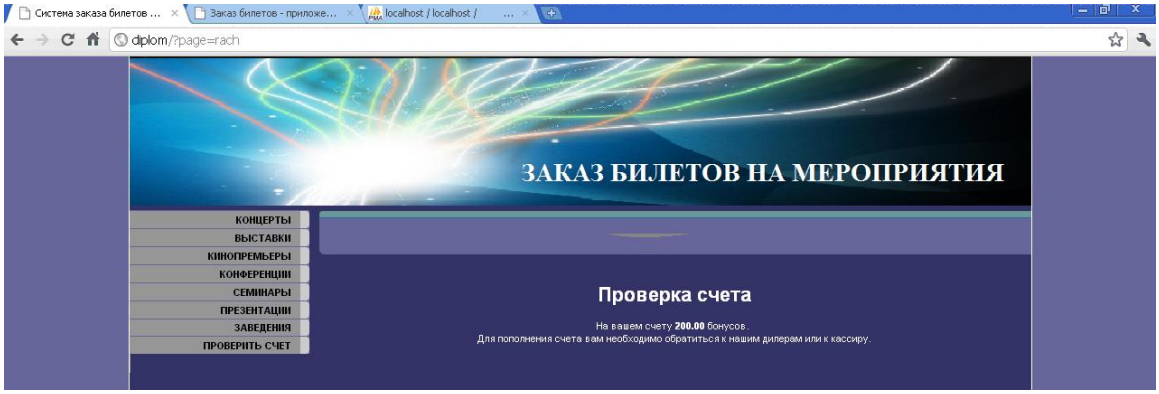


Рисунок 4.13 –Повідомлення о наявної кількості бонусів на рахунку клієнта

За допомогою касира клієнтом може бути поповнено особовий рахунок: касир отримує готівкою грошові кошти и зараховує бонуси на рахунок клієнта.

4.4 Керівництво адміністратора системи

При здійсненні програмної реалізації інформаційної розподіленої системи замовлення квитків передбачено додаток для адміністрування створених веб-сервісів. Здійснює цю процедуру у системі Користувач-Адміністратор. Безпосередньо адміністратор системи має таки права: аутентифікації та авторизація різних категорій користувачів системи; здійснення додавання, зміни, редагування, видалення заходів, закладів, типів заходів, користувачів, додання інформаційної підтримки та анонсування майбутніх запланованих підприємствами-організаторами культурно-масових заходів. Для здійснення процедури авторизації користувачів в системі, передбачена реєстраційна форма (рис. 4.14).

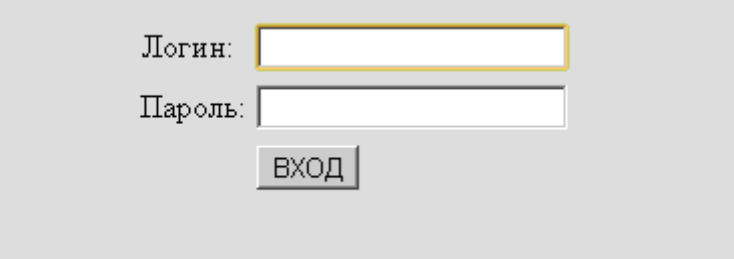


Рисунок 4.14 – Форма авторизації Касира та Адміністраторів системи

Для здійснення додавання нових культурних заходів у сервісі замовлення квитків для Адміністратора передбачена форма (рис. 4.15).

Рисунок 4.15 – Форма Адміністратора для роботи з заходами

За допомогою полів запропонованої форми здійснюється додавання всієї необхідної інформації о заходах. Передбачені поля: назва, дата проведення, додаткова інформація, тип заходу, місце проведення, необхідні

зображення. Після натискання кнопки Записати, здійснюється внесення даних у таблицю.

Реалізована функція відображення всіх наявних заходів у таблиці: назва заходу, дата проведення. Також передбачена можливість здійснення не тільки перегляду але й видалення внесених раніше заходів або їх редагування. Якщо Адміністраторові необхідно видалити сторінку, то при видаленні існуючого об'єкта інтерфейс адміністратора запитує підтвердження для здійснення такої операції для захисту від помилкового видалення.

Також в сервісі замовлень квитків на культурно-масові заходи передбачено додавання нових типів запланованих культурних заходів. Передбачені поля: назва нового типу (наприклад, концерт, семінар чи будь що), дата проведення, додаткова інформація, тип заходу, місце проведення, необхідні зображення.

В рамках реалізації системи замовлень квитків реалізовано веб-сервіс для роботи Користувача-Касира. Сервіс-орієнтована підсистема для касира реалізує наступні функції: додавання інформації про облік і продаж квитків і здійснення керування бонусними особистими рахунками Зареєстрованих клієнтів, внесення даних про нарахування бонусів по рахунках на підставі здійснення оплати готівкою у касі продажу квитків.

Для категорії користувача Касир, система передбачає реалізацію наступних прав: аутентифікація та авторизація в системі, перегляд замовлень користувачів і їх зміна, пошук необхідного замовлення, можливість здійснення операції поповнення рахунку, реєстрація оплати Касиром квитків, придбаних клієнтами на заходи (рис. 4.16).

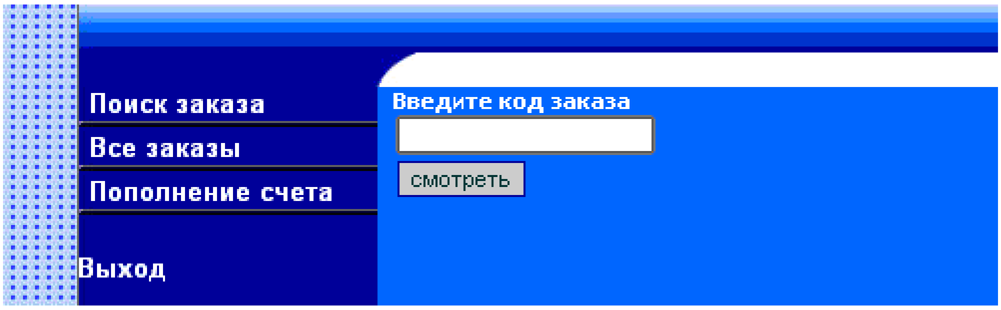


Рисунок 4.16 – Форма реєстрації Користувача-Касир у системі

Користувач-Касир може здійснювати перегляд всього списку замовлень, які зробили різні користувачі (рис. 5.17).

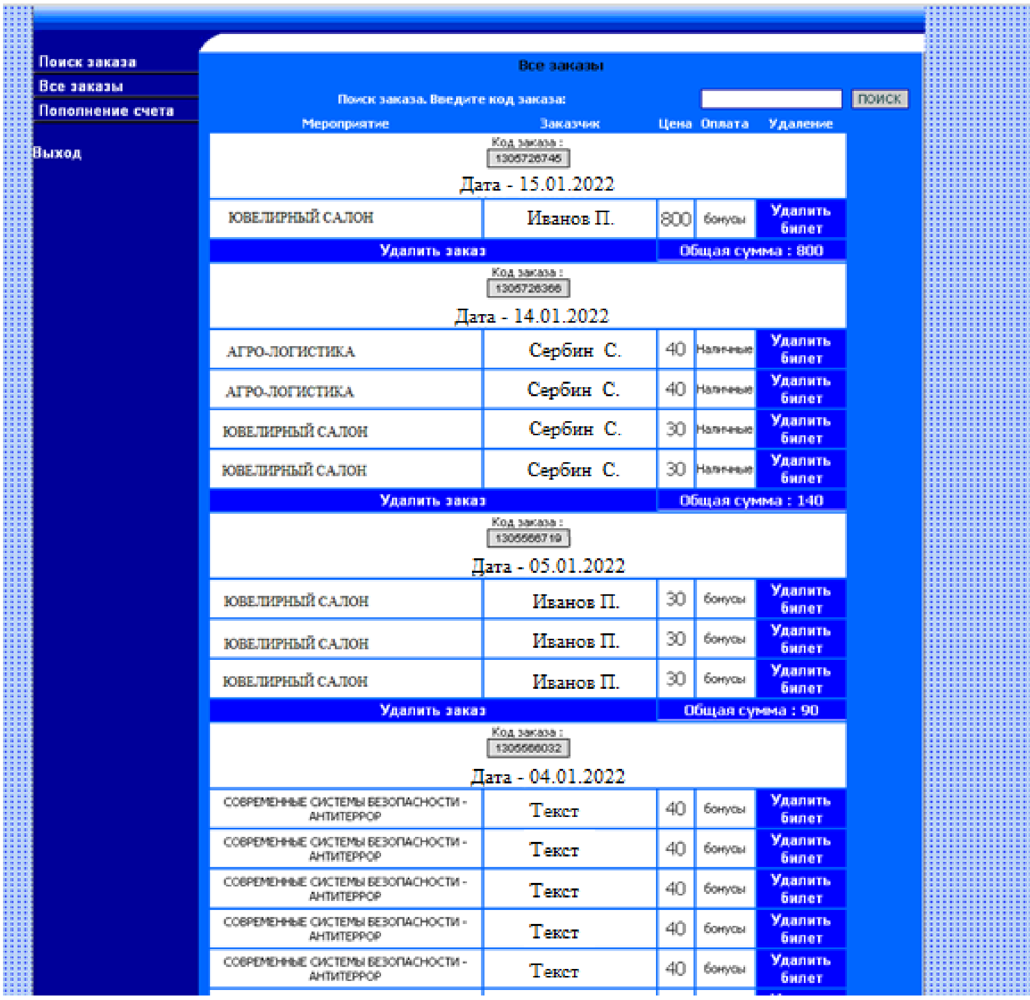


Рисунок 5.17 – Форма перегляду списку замовлень

В системі передбачена можливість пошуку із існуючого списку замовлень необхідного замовлення. Здійснення пошуку відбувається за номером замовлення, який необхідно ввести у вікно пошуку.

Для здійснення поповнення персональних рахунків користувачів передбачена форма поповнення бонусного рахунку. Здійснити цю операцію має право користувач Касир, за допомогою реалізованої у системі функції (рис. 5.18).

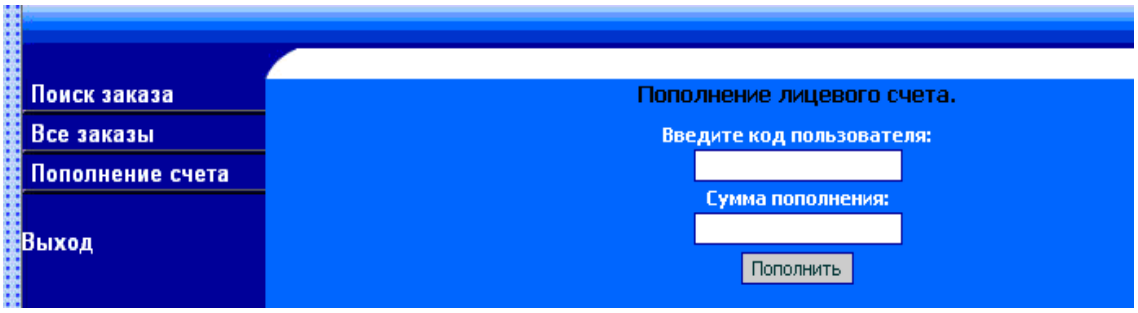


Рисунок 5.18 – Форма поповнення особистих бонусних рахунків

Доступ до web-сервісу здійснюється по протоколу HTTP, а обмін даними відбувається в форматі XML. Тому можна стверджувати, розробка web-послуг в обраній предметній області є актуальним завданням. Основним призначенням розробленого сервісу замовлення квитків на культурно-масові заходи, заснованого на використанні сервіс-орієнтованої технології, є допомога web-ресурсам організаторів культурно-масових заходів в наданні як інформації для їх користувачів та і наданні можливості замовлення квитків засобами залучання розроблених веб-сервісів на своїх інформаційних ресурсах, який функціонує у мережі Інтернет. Тобто розроблений сервіс спроектовано і функціонує так, що їм можуть користуватись не звичайні відвідувачі мережі Internet, а розробники чи адміністратори вже існуючих web-ресурсів, які зацікавлені в цій інформації, що надає розроблений web-сервіс.

ВИСНОВКИ

В результаті виконання дипломної роботи було створено web-сервіс служби продажу квитків засобами сервіс-орієнтованої технології, що дозволило застосовувати сервіс для підтримки діяльності установ задіяних в організації культурно-масових заходів. Основним призначенням сервісу є залучення клієнтів, інформування зацікавлених осіб про майбутні культурно-масові заходи різних видів (виставки, концерти, конференції і т.п.) та надання можливості здійснення замовлень квитків засобами мережі Інтернет.

Основу розробленого web-сервісу складають платформенно-незалежні WWW технології, тому вони володіють незалежністю від програмного забезпечення, на якому реалізовані сервіси, і від платформи, на якій вони функціонують. Додатки, створені для різних типів користувачів системи, застосовують лише результат роботи web-сервісу.

В ході виконання дипломної роботи проведено аналіз предметної області, сформульовані вимоги до функцій сервісу, здійснено вибір і обґрунтування сервіс-орієнтованої архітектури та програмних засобів реалізації, проведено проектування та моделювання бізнес-процесів даної предметної області з застосуванням методології ARIS, здійснено програмну реалізацію сервісу. Розроблений web-сервіс може бути використано в якості будівельних блоків для будь-яких розподілених web-систем у мережі Інтернет. Розроблений сервіс є інструментом для розробників або адміністраторів будь-яких web-систем, що зацікавлені в інформації, яку надає цей web-сервіс.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Николайчук Я.М., Возна Н.Я., Пітух І.Р. Проектування спеціалізованих комп’ютерних систем / Навчальний посібник / Тернопіль: ТзОВ "Терно граф", 2010, 382 с.
2. Лотоцька М.Р. Інформаційні системи і технології в управлінні організацією. Курс лекцій. Прикарпатський національний університет імені В.Стефаника. Івано-Франківськ, 2013, 278 с.
3. Гужва В.М. Інформаційні системи і технології на підприємствах: навч. посіб. К. : КНЕУ, 2001, 400 с.
4. О.В. Суботін. Конспект лекцій «Розподілені комп'ютерні системи та мережі», Краматорськ: ДДМА, 2017, 52 с.
5. Rosen M. Applied SOA: Service-Oriented Architecture and Design Strategies / Mike Rosen, Boris Lublinsky, Kevin T. Smith, Narc J. Balcer //Wiley Publishing, Inc. 2008, 699p.
6. Підручник з хмарних обчислень для початківців. URL: https://www.guru99.com/cloud-computing-for-beginners.html (дата звернення: 02.04.2022).
7. Алекс Феррара, Мєтью Макдональд. Программирование web-сервисов для .NET. / Пер. с англ. СПб.: Питер, 2003, 429 с.
8. Томасон Лаура. Разработка Web-приложений на PHP и MySQL: Пер. с анг./ Лаура Томсон, Люк Веллинг. 2-е изд., испр. – СПб.: ООО "ДиаСофтЮП", 2003, 672 с.
9. Береза А.М.. Основи створення інформаційних систем: Навч. посібник. К.: КНЕУ, 2001, 214с.
10. С. В. Мінухін, О. М. Беседовський, С. В. Знахур. Методи і моделі проектування на основі сучасних CASE-засобів. Навчальний посібник. Х: Вид. ХНЕУ, 2008, 272 с.
11. Грейвс М. Проектирование баз данных на основе XML. – М.-СПб-К.: «Вильямс», 2002, 640 с.

12. Косентино Кристофер. PHP Web-профессионалам. / Пер. с англ. Киев: BHV, 2001, 206 с.

ДОДАТОК А Програмні коди основних модулів

Скрипт створення бази даних системи

```
CREATE TABLE `ACTION` (
  id int(11) unsigned not null auto_increment,
  act_date int(11) not null default '0',
  `name` text not null,
  info text not null,
  id_inst int(11) not null default '0',
  type_action int(11) not null default '0',
  sponsors text not null,
  img varchar(255) not null default '',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE ADMINS (
  id int(11) unsigned not null auto_increment,
  login varchar(255) not null default '',
  pass varchar(255) not null default '',
  `name` varchar(255) not null default '',
  type_adm int(2) not null default '0',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE ADMIN_TYPE (
  id int(11) unsigned not null auto_increment,
  `name` varchar(255) not null default '',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE COST (
  id int(11) unsigned not null auto_increment,
  id_action int(11) not null default '0',
  id_zone int(11) not null default '0',
  cost varchar(255) not null default '',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE INSTITUTION (
  id int(11) unsigned not null auto_increment,
  `name` varchar(255) not null default '',
  adres varchar(255) not null default '',
  tel varchar(255) not null default '',
  url varchar(255) not null default '',
  opis text,
  img varchar(255) not null default '',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE ORDERS (
  id int(11) unsigned not null auto_increment,
```

```
  id_user int(11) default null,
  ord_data int(11) not null default '0',
  kodes varchar(255) not null default '0',
  card_or_kesh int(2) not null default '0',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE PLACE (
  id int(11) unsigned not null auto_increment,
  id_zone int(11) not null default '0',
  n_rad int(11) not null default '0',
  n_place int(11) not null default '0',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE TYPE_ACTION (
  id int(10) unsigned not null auto_increment,
  `name` varchar(255) not null default '',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE USERS (
  id int(11) unsigned not null auto_increment,
  `first` varchar(255) not null default '',
  `last` varchar(255) not null default '',
  patro varchar(255) default null,
  e_mail varchar(255) default null,
  tel varchar(255) default null,
  kod varchar(255) not null default '',
  number decimal(11,2) unsigned not null default '0.00',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE USER_ORD (
  id int(11) unsigned not null auto_increment,
  id_ord int(11) not null default '0',
  id_action int(11) not null default '0',
  id_place int(11) not null default '0',
  primary key (id)
) engine=myisam  default charset=utf8;

CREATE TABLE ZONE (
  id int(11) unsigned not null auto_increment,
  id_inst int(11) not null default '0',
  `name` varchar(255) not null default '',
  primary key (id)
) engine=myisam  default charset=utf8;
```

soapserver.php

```
<?php
     $dbhost = "localhost"; // Имя хоста БД
     $dbusername = "webservice"; // Пользователь БД
     $dbpass = "123"; // Пароль к базе
     $db = "football"; // Имя базы
     $dbconnect = mysql_connect ($dbhost, $dbusername, $dbpass);
     mysql_set_charset( 'utf8' );//кодировка
     if (!$dbconnect) { echo ("Не могу подключиться к серверу
базы данных!"); return 0; exit();}
     if(!@mysql_select_db($db)) {die ("Не могу подключиться к
базе данных!"); return 0; exit();}
```

```
     * функция возвращает список предстоящих мероприятий
     *
     * @return string $arr
     */
     function getEvents(){
          $sql = "SELECT id_event,price,title,date
                    FROM `event`
                    WHERE date >= curdate();
                    ";
                $result = mysql_query($sql);
                $row1 = array();

                while($row = mysql_fetch_array($result)){
                     $row1[] = $row;
                }
                $arr = serialize($row1);
                     return $arr;
     }


     * @param string $param (имя, email, телефон, идентификатор
мероприятия, кол-во билетов)
     *
     * @return boolean
     */
     function bron($param){
          $pArr = array();
          $pArr = unserialize($param);

          $name = strip_tags($pArr['name']);
         $phone = strip_tags($pArr['phone']);
         $email = strip_tags($pArr['email']);
         $kol_vo = strip_tags($pArr['kol_vo']);
         $id_event = strip_tags($pArr['id_event']);
          $to = "admin@mail.ru";
         $theme_email = "НОВЫЙ ЗАКАЗ НА БИЛЕТЫ";

         $res = mysql_query("SELECT `title`, `description` FROM
`event` WHERE `id_event` = $id_event");
         $row = mysql_fetch_array($res);

         $message = "Заказ на событие \"".$row[title]."\". \n
Описание события:\n".$row[description].".";

          $mail = mail( $to, $theme_email,"
             ФИО: $name\n
             Номер телефона: $phone\n
             Адрес: $email\n
             Количество билетов: $kol_vo\n
             Событие: $message\n",
             "Content-type: text/plain; charset=utf-8");
          $result = mysql_query("INSERT INTO `bron`(`name`,
`email`, `phone`, `kol_vo`, `date`, `id_event`) VALUES ('$name',
'$email', '$phone', '$kol_vo', curdate(), '$id_event')");
          if ($result){ return TRUE;}
          else{return FALSE;}
     }}
          ini_set("soap.wsdl_cache_enabled", 0);
     //создаем soap сервер
```

```
     $server = new
SoapServer("http://webservice/wsdl/football.wsdl");
     //добавяем класс к серверу
     $server->SetClass("FootballService");
     //Запускаем сервер
     $server->handle();

                              soapclient.php
<?
     try{
          //создание клиента
          $client = new
SoapClient("http://webservice/wsdl/football.wsdl"/*,array(
         "trace"      => 1,
         "exceptions" => 0)*/);
          //посылка soap-запроса и получение результата
          $id = 1;
          $result = $client->getClubById($id);
          $res = unserialize($result);
          //вывод результата
          /*print_r($result);*/
          //print $object->$result['id'];
          print "<pre>\n";
          /*print "Запрос :\n".htmlspecialchars($client-
>__getLastRequest()) ."\n";
     print "Ответ:\n".htmlspecialchars($client-
>__getLastResponse())."\n";*/
          print_r($res);
          print "</pre>";
     }
     catch(SoapFault $exception) {
          echo $exception->getMessage();
     }
?>
kwiks.js
 (function($){
     $.fn.kwicks = function(options) {
          var defaults = {
               isVertical: false,
               sticky: false,
               defaultKwick: 0,
               event: 'mouseover',
               spacing: 0,
               duration: 500
          };
          var o = $.extend(defaults, options);
          var WoH = (o.isVertical ? 'height' : 'width'); // WoH
= Width or Height
          var LoT = (o.isVertical ? 'top' : 'left'); // LoT =
Left or Top
                    return this.each(function() {
               container = $(this);
               var kwicks = container.children('li');
               var normWoH =
kwicks.eq(0).css(WoH).replace(/px/,''); // normWoH = Normal
Width or Height
               if(!o.max) {
                    o.max = (normWoH * kwicks.size()) - (o.min *
(kwicks.size() - 1));
               } else {
```

```
                    o.min = ((normWoH * kwicks.size()) - o.max) /
(kwicks.size() - 1);
               }
               // set width of container ul
               if(o.isVertical) {
                    container.css({
                         width : kwicks.eq(0).css('width'),
                         height : (normWoH * kwicks.size()) +
(o.spacing * (kwicks.size() - 1)) + 'px'
                    });
               } else {
                    container.css({
                         width : (normWoH * kwicks.size()) +
(o.spacing * (kwicks.size() - 1)) + 'px',
                         height : kwicks.eq(0).css('height')
                    });
               }
               // pre calculate left or top values for all
kwicks but the first and last
               // i = index of currently hovered kwick, j =
index of kwick we're calculating
               var preCalcLoTs = []; // preCalcLoTs = pre-
calculated Left or Top's
               for(i = 0; i < kwicks.size(); i++) {
                    preCalcLoTs[i] = [];
                    // don't need to calculate values for first
or last kwick
                    for(j = 1; j < kwicks.size() - 1; j++) {
                         if(i == j) {
                              preCalcLoTs[i][j] = o.isVertical ?
j * o.min + (j * o.spacing) : j * o.min + (j * o.spacing);
                         } else {
                              preCalcLoTs[i][j] = (j <= i ? (j *
o.min) : (j-1) * o.min + o.max) + (j * o.spacing);
                         }
                    }
               }
               // loop through all kwick elements
               kwicks.each(function(i) {
                    var kwick = $(this);
                    // set initial width or height and left or
top values
                    // set first kwick
                    if(i === 0) {
                         kwick.css(LoT, '0px');
                    }
                    // set last kwick
                    else if(i == kwicks.size() - 1) {
                         kwick.css(o.isVertical ? 'bottom' :
'right', '0px');
                    }
                    // set all other kwicks
                    else {
                         if(o.sticky) {
                              kwick.css(LoT,
preCalcLoTs[o.defaultKwick][i]);
                         } else {
                              kwick.css(LoT, (i * normWoH) + (i *
o.spacing));
                         }
```

```
                    }
                    // correct size in sticky mode
                    if(o.sticky) {
                         if(o.defaultKwick == i) {
                              kwick.css(WoH, o.max + 'px');
                              kwick.addClass('active');
                         } else {
                              kwick.css(WoH, o.min + 'px');
                         }
                    }
                    kwick.css({
                         margin: 0,
                         position: 'absolute'
                    });

                    kwick.bind(o.event, function() {
                         // calculate previous width or heights
and left or top values
                         var prevWoHs = []; // prevWoHs =
previous Widths or Heights
                         var prevLoTs = []; // prevLoTs =
previous Left or Tops
                         kwicks.stop().removeClass('active');
                         for(j = 0; j < kwicks.size(); j++) {
                              prevWoHs[j] =
kwicks.eq(j).css(WoH).replace(/px/, '');
                              prevLoTs[j] =
kwicks.eq(j).css(LoT).replace(/px/, '');
                         }
                         var aniObj = {};
                         aniObj[WoH] = o.max;
                         var maxDif = o.max - prevWoHs[i];
                         var prevWoHsMaxDifRatio =
prevWoHs[i]/maxDif;
                         kwick.addClass('active').animate(aniObj,
{
                              step: function(now) {
                                   // calculate animation
completeness as percentage
                                   var percentage = maxDif != 0 ?
now/maxDif - prevWoHsMaxDifRatio : 1;
                                   // adjsut other elements based
on percentage
                                   kwicks.each(function(j) {
                                        if(j != i) {

     kwicks.eq(j).css(WoH, prevWoHs[j] - ((prevWoHs[j] - o.min)
* percentage) + 'px');
                                        }
                                        if(j > 0 && j <
kwicks.size() - 1) { // if not the first or last kwick

     kwicks.eq(j).css(LoT, prevLoTs[j] - ((prevLoTs[j] -
preCalcLoTs[i][j]) * percentage) + 'px');
                                        }
                                   });
                              },
                              duration: o.duration,
                              easing: o.easing
                         });
```

```
                    });
               });
               if(!o.sticky) {
                    container.bind("mouseleave", function() {
                         var prevWoHs = [];
                         var prevLoTs = [];
                         kwicks.removeClass('active').stop();
                         for(i = 0; i < kwicks.size(); i++) {
                              prevWoHs[i] =
kwicks.eq(i).css(WoH).replace(/px/, '');
                              prevLoTs[i] =
kwicks.eq(i).css(LoT).replace(/px/, '');
                         }
                         var aniObj = {};
                         aniObj[WoH] = normWoH;
                         var normDif = normWoH - prevWoHs[0];
                         kwicks.eq(0).animate(aniObj, {
                              step: function(now) {
                                   var percentage = normDif != 0
? (now - prevWoHs[0])/normDif : 1;
               for(i = 1; i < kwicks.size(); i++) {
kwicks.eq(i).css(WoH, prevWoHs[i] - ((prevWoHs[i] - normWoH) *
percentage) + 'px');
          if(i < kwicks.size() - 1) {
kwicks.eq(i).css(LoT, prevLoTs[i] - ((prevLoTs[i] - ((i *
normWoH) + (i * o.spacing))) * percentage) + 'px');
                              }
                                   }
                              },
                              duration: o.duration,
                              easing: o.easing
                         });
                    });
               }
          });
     };
})(jQuery);

lock.php
<?php
include("blocks/connect.php");
if (!isset($_SERVER['PHP_AUTH_USER']))
{
        Header ("WWW-Authenticate: Basic realm=\"Admin Page\"");
        Header ("HTTP/1.0 401 Unauthorized");
        exit();
}
else {
        if (!get_magic_quotes_gpc()) {
                $_SERVER['PHP_AUTH_USER'] =
mysql_escape_string($_SERVER['PHP_AUTH_USER']);
                $_SERVER['PHP_AUTH_PW'] =
mysql_escape_string($_SERVER['PHP_AUTH_PW']);
        }
        $query = "SELECT userPass FROM userlist WHERE
userName='".$_SERVER['PHP_AUTH_USER']."'";
        $lst = @mysql_query($query);
        if (!$lst)
        {
```

```
            Header ("WWW-Authenticate: Basic realm=\"Admin
Page\"");
            Header ("HTTP/1.0 401 Unauthorized");
            exit();
        }
        if (mysql_num_rows($lst) == 0)
        {
           Header ("WWW-Authenticate: Basic realm=\"Admin
Page\"");
           Header ("HTTP/1.0 401 Unauthorized");
           exit();
        }
        $pass =  @mysql_fetch_array($lst);
        if ($_SERVER['PHP_AUTH_PW']!= $pass['pass'])
        {
           Header ("WWW-Authenticate: Basic realm=\"Admin
Page\"");
           Header ("HTTP/1.0 401 Unauthorized");
           exit();
        }}?>
```