

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ПРОГРАМНОЇ
ІНЖЕНЕРІЇ

Інформаційна система «Аукціон»

Курсовий проект з дисципліни «Бази даних»

Виконавці
здобувачі ВО гр. ПІ-181



Павлюк В.В.,
Прищеп Д.О.

Керівник
викладач

Войцеховська М.М.

Я, Гавриш В'ячеслав Віталійович, підтверджую, що дана робота є моєю власною письмовою роботою, оформленою з дотриманням цінностей та принципів етики і академічної доброчесності відповідно до Кодексу академічної доброчесності Національного університету «Чернігівська політехніка». Я не використовував/ла жодних джерел, крім процитованих, на які надано посилання в роботі.

17.05.2021

ТБС-

Дата

Підпис

Я, Грицина Ларина Олександрівна, підтверджую, що дана робота є моєю власною письмовою роботою, оформленою з дотриманням цінностей та принципів етики і академічної доброчесності відповідно до Кодексу академічної доброчесності Національного університету «Чернігівська політехніка». Я не використовував/ла жодних джерел, крім процитованих, на які надано посилання в роботі.

17.05.2021

Дата


Підпис

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи з дисципліни «Бази даних»

Павлюк В.В. , гр. ІІІ-181

Прищеп Д.О. , гр. ІІІ-181

Тема роботи: БД "Аукціон"

Передбачувані технічні та експлуатаційні результати роботи:

Програмна система «Аукціон» являє собою корпоративний додаток, який дозволить проводити торги онлайн та отримувати повідомлення про їх результати, а також керувати базою даних через зручний інтерфейс.

Розробка БД повинна включати:

а) побудову концептуальної моделі предметної області, що включає не менш 3-х об'єктів, пов'язаних між собою взаємозв'язками типу 1: М, у вигляді діаграми «сутність-зв'язок»;

б) побудову нормалізованої логічної моделі предметної області (не менше 3-х таблиць) із зазначенням: первинних і зовнішніх ключів, типів даних атрибутів;

в) реалізацію на рівні структури БД засобів забезпечення цілісності даних: унікальність і обов'язковість введення первинних ключів; підтримка посилювальної цілісності для зовнішніх ключів; значення атрибутів за замовчуванням і обов'язковість введення заданих атрибутів;

г) реалізацію базових запитів до БД на пошук і додавання даних;

д) реалізацію бізнес-логіки зі сторони серверу БД: тригерів, збережених процедур та представлень;

є) реалізацію розділеного доступу до БД зі сторони серверу БД: користувачі та їх права;

ж) реалізацію надійності збереження даних: механізми реплікації та кластеризації даних.

Взаємодія з БД має бути реалізована в архітектурі клієнт-сервер з наступними відокремленими рівнями:

1) бази даних;

- 2) доступу до даних з використанням технологій об'єктно-реляційного маппінгу;
- 3) обробки даних (бізнес-логіки);
- 4) сервісів;
- 5) інтерфейсу.

Обсяг текстової та графічної документації

Пояснювальна записка до проекту обсягом 35-50 сторінок друкованого тексту формату А4 і програмна документація на систему обсягом 25-35 сторінок друкованого тексту формату А4. Обсяги текстової інформації можуть бути скориговані в процесі роботи за погодженням з керівником.

Планові терміни по етапах:

№	Назва етапів роботи	Термін виконання	Примітки
1	Узгодження завдання	01.03.2021	
2	Аналіз предметної області. Розробка БД	15.03.2021	
3	Прототип призначеного для користувача інтерфейсу	31.03.2021	
4	Модулі та бізнес логіка	20.04.2021	
5	Інтерфейс користувача	04.05.2021	
6	Оформлення роботи	11.05.2021	
7	Демонстрація працездатності програмного проекту	07.05.2021	
8	Плановий термін захисту проекту	17.05.2021	

Керівник роботи: _____ Войцеховська М.М.



Виконавці роботи: _____ Павлюк В.В.

_____ Прищепа Д.О.

Дата видачі завдання

«10» лютого 2021 р

СПИСОК АВТОРІВ

ІІБ співавтора	ОПИС ЧАСТИНИ РОБОТИ	ПІДПИС
ЗВО гр. ПІ-181 Павлюк В.В.	<p>Проектування та реалізація бази даних, в т.ч. бізнес-логіки.</p> <p>Проектування та реалізація шару доступу до даних.</p> <p>Оформлення звіту: вступ; реферат; розділ 1; розділ 2.1, 2.2, 2.3.1, 2.3.2; розділ 3.1, 3.2.1, 3.2.2.</p>	
ЗВО гр. ПІ-181 Прищєпа Д.О.	<p>Проектування діаграми використання системи.</p> <p>Вибір засобів розробки та проектування архітектури системи.</p> <p>Проектування та реалізація шару сервісів</p> <p>Проектування та реалізація шару відображення.</p> <p>Оформлення звіту: розділ 2.3, 2.3.4; розділ 3.2, 3.2.3, 3.2.4; висновки.</p>	

РЕФЕРАТ

Курсова робота, 89 с., 61 рисунок, 7 таблиць, 6 джерел, 1 додаток

Мета розробки курсової роботи — реалізувати багаторівневу систему, яка дозволить проводити торги онлайн та отримувати повідомлення про їх результати, а також керувати базою даних через зручний інтерфейс.

Розробка та управління базою даних здійснена за допомогою СКБД PostgreSQL 10. Для проектування модулів програми використано UML-діаграми, пояснювальні таблиці, макети. Додаток був реалізований на таких мовах: SQL, HTML, CSS, JS, JAVA, HQL, JSTL, EL. Також були використані наступні технології для взаємодії з даними: ORM, JavaxMail, Servlets, JSP, Google Visualization API, Bootstrap.

Додаток «Аукціон» реалізує наступні функції:

- відображення усіх активних лотів;
- можливість зробити ставку;
- можливість стати продавцем чи покупцем;
- сповіщення продавців та учасників торгу про кінцевого покупця;
- можливість видалення лоту, ставки тощо;
- контроль над ставками та їх редагуванням;
- збір статистики та моніторинг аукціону;
- можливість сортування лотів за критеріями.

Результат розробки та реалізації програми представлений у вигляді пояснювальної записки до проекту.

Робота програми можлива при наявності 256 Мб оперативної пам'яті, процесора не нижче Intel Pentium 4 або AMD Athlon.

Для роботи клієнтської частини додатку необхідна наявність браузера Google Chrome версії не нижче 1.0.

КЛЮЧОВІ СЛОВА: СКБД, АУКЦІОН, JAVA, POSTGRESQL, HIBERNATE, SQL, БАГАТОРІВНЕВА СИСТЕМА, КЛІЄНТ, СЕРВЕР.

THE ABSTRACT

Course project, 89 pages, 61 figures, 7 table, 6 sources, 1 appendix

The purpose of the course work is to implement a multi-level system that will allow online bidding and receive notifications about their results, as well as manage the database through a user-friendly interface.

Development and management of the database is carried out using DBMS PostgreSQL 10. For the design of program modules used UML-diagrams, explanatory tables, layouts. The application was implemented in the following languages: SQL, HTML, CSS, JS, JAVA, JSTL, EL. The following technologies were also used to interact with the data: ORM, Servlets, JSP, Google Visualization API, Bootstrap, presentation mechanism (View).

The "Auction" application implements the following functions:

- display of all active lots;
- opportunity to place a bet;
- the opportunity to become a seller or buyer;
- notification of sellers and bidders about the final buyer;
- the ability to delete the lot, bid, etc .;
- control over bids and their editing;
- collection of statistics and monitoring of the auction;
- the ability to sort lots by criteria.

The result of the development and implementation of the program is presented in the form of an explanatory note to the project.

The program is possible with 256 MB of RAM, processor not lower than Intel Pentium 4 or AMD Athlon.

The client part of the application requires a Google Chrome browser version at least 1.0.

KEY WORDS: DBMS, AUCTION, JAVA, POSTGRESQL, HIBERNATE, SQL, MULTILEVEL SYSTEM, CLIENT, SERVER.

ЗМІСТ

ВСТУП	9
1 АНАЛІЗ ВИРІШУВАНОЇ ЗАДАЧІ	11
1.1 Опис предметної області	11
1.2 Аналіз предметної області.....	11
1.3 Мета і завдання системи.....	14
1.4 Призначення системи.....	14
1.5 Вимоги до системи	17
1.6 Висновки до розділу 1	17
2 ПРОЕКТУВАННЯ СИСТЕМИ.....	18
2.1 Проектування бази даних	18
2.1.1 Опис таблиць бази даних.....	18
2.1.2 Проектування логічної схеми БД.....	20
2.1.3 Проектування бізнес-логіки та бізнес-правил	21
2.2 Проектування архітектури системи.....	29
2.3 Вибір інструментальних засобів розробки системи.....	30
2.3.1 Сервер баз даних	30
2.3.2Технології реалізації системи	33
2.4 Проектування модулів системи	35
2.4.1 Проектування модулю доступу до даних	35
2.4.4 Проектування шару відображення	39
2.5 Висновки до розділу 2	43
2 РОЗРОБКА СИСТЕМИ	44
3.1Розробка бази даних системи.....	44
3.1.1 Розробка фізичної схеми бази даних.....	44
3.1.2Забезпечення цілісності даних	44
3.1.4 Перевірка ефективності доступу до даних	51
3.1.5 Підвищення надійності доступу до даних	53
3.2 Розробка модулів системи.....	54
3.2.1 Розробка модулів шару бізнес логіки і бізнес правил	54
3.2.3 Розробка шару сервісів	57
3.3 Тестування створеної системи	60
3.4 Висновки до розділу 3	64
ВИСНОВКИ.....	65
ВИКОРИСТАНІ ДЖЕРЕЛА	66
ДОДАТКИ.....	67

ВСТУП

Проектування будь-якого додатку починається з питання збереження та обробки інформації про товари, користувачів, послуги тощо. І зі збільшенням потоку даних для збереження ця проблема стає все актуальнішою. Виникають труднощі із швидким пошуком необхідної інформації та її обробкою, сортуванням цього розмаїття цифр, прізвищ, імен та ін.

Одним з найпопулярніших методів вирішення поставлених задач є використання БД. База даних (БД) — це організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, переважно великих обсягів. Ця структура зберігає потік даних у вигляді взаємозалежних таблиць, що значно спрощує роботу майбутнього ПЗ. Щоб створити запит до бази даних часто використовують Structured Query Language. SQL дає змогу додавати, редагувати та видаляти інформацію, що міститься у таблицях[1].

Майже кожний сучасний веб-сервіс, створює власну БД для зберігання необхідних даних та керує нею за допомогою спеціальних системи управління базами даних, прикладом яких є PostgreSQL[3], MySQL, Oracle Database тощо. Такі системи управління відрізняються централізованою обробкою запитів, забезпечують надійність, доступність та безпеку БД[1].

Не винятком серед таких сервісів є і онлайн-аукціони. Сучасний ринок веб-торгів заповнений великою кількістю додатків, сайтів, систем, де кожний користувач може взяти участь у купівлі бажаного лоту, однак орієнтуватися у такому ПЗ складно через громіздкий інтерфейс, «ручний» моніторинг торгів, відсутній або неактуальний функціонал для сортування лотів, складність керування виставленими товарами тощо.

Отже, поліпшенню ситуації на ринку сприятиме створення інформаційної системи «Аукціон», яка надасть користувачам можливість

орієнтуватися у інтерфейсі на рівні інтуїтивного розуміння, буде відсилати сповіщення учасникам торгів про кінцевого покупця, дозволить виконувати сортування за актуальними на сьогоднішній день критеріями, розмежує доступ для користувачів та спростить керування виставленими лотами.

1 АНАЛІЗ ВИРІШУВАНОЇ ЗАДАЧІ

1.1 Опис предметної області

Створили опис предметної області з метою розуміння механіки проведення торгів, оскільки необхідно створити систему онлайн-аукціону, що являє собою спеціально організований і періодично діючий ринок продажу лотів, майна з публічного торгу покупцеві, який запропонував найвищу ціну.

Лот – це одиниця купівлі-продажу під час торгів на аукціонах, біржах. Розміром лота відповідає визначений наперед обсяг товару в натуральному вираженні. Стандартний розмір угоди, контракту, що здійснюються під час торгів, встановлюється правилами аукціонної і біржової торгівлі. Під час торгів на аукціоні виставляється на продаж лот, який складається з одного або декількох однорідних за якістю предметів, речей, наборів. Кожному аукціонному лоту присвоюється порядковий номер і встановлюється своя аукціонна ціна в ході торгу. Джерелом інформації в цьому випадку стали загальні відомості про реальні аукціони.

1.2 Аналіз предметної області

На основі складеного опису створили модель предметної області (рисунк 1.1), основними елементами якої стали:

- item (Товар);
- buyer (Покупець);
- bid (Ставка);
- seller (Продавець);
- category (Категорія);
- lot (Лот).

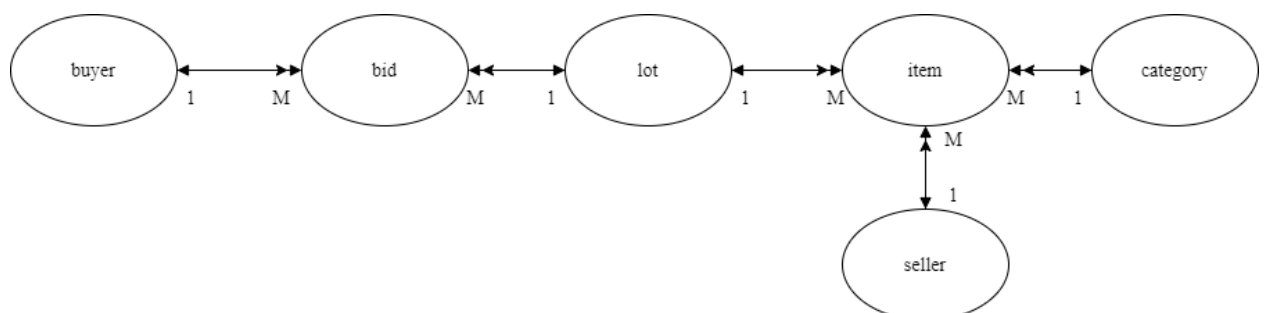


Рисунок 1.1 — Модель предметної області «Аукціон»

Зі створеної моделі можна побачити, що предметна область складається із шести виділених раніше сутностей, які пов'язані між собою зв'язком типу «1:М». Такий зв'язок означає, що екземпляри однієї сутності(позначені символом «1») відповідають декільком записам іншої(позначені символом «М»). Наприклад, один покупець може зробити декілька ставок. Для кожної сутності визначили первинний ключ(код), а також неключові атрибути:

- Лот. Атрибутами сутності лот стали: <код>, <початкова ціна>, <початок торгу>, <кінець торгу>, <поточна ціна>, <статус лоту>, де первинним ключем є <код>.

- Ставка. Атрибутами сутності ставка стали: <код>, <код лоту>, <код покупця>, <ціна>, <статус>, <час ставки>, де первинним ключем є <код>. Зовнішніми ключами стали атрибути <код лоту> та <код покупця>.

- Покупець. Атрибутами сутності покупець стали: <код>, <ім'я>, <електронна пошта>, <адреса>, <пароль>, <номер телефону>, де первинним ключем є <код>.

- Категорія. Атрибутами сутності категорія стали: <код>, <назва категорії>, <головна категорія>, де первинним ключем є <код>.

- Товар. Атрибутами сутності товар стали: <код>, <код продавця>, <код категорії>, <код лоту>, <назва товару>, <опис товару>, де первинним ключем є <код>. Зовнішніми ключами стали атрибути <код продавця> та <код категорії>.

- Продавець. Атрибутами сутності проданий товар стали: <код>, <ім'я продавця>, <електронна пошта продавця>, <пароль продавця>, <телефонний номер продавця>.

З огляду на предметну область та реальний світ <номер телефону> має складати десять цифр, а <електронна пошта> — знак '@' а також '.', також такі поля як <електронна пошта>, <назва товару>, <назва категорії>, <номер телефону> мають бути унікальними.

Склали ряд істинних висловлювань на основі моделі предметної області для визначення параметрів між об'єктами сутностей:

- кожний Покупець може зробити одну або кілька ставок;
- кожна Ставка може мати лише одного Покупця;
- кожний Лот може мати нуль, одну або кілька Ставок;
- кожна Ставка може бути здійснена лише на один Лот;
- кожний Лот може складатися з декількох Товарів;
- кожний Товар може входити лише до одного Лоту;
- кожна Категорія може включати до себе нуль, один або декілька Товарів;
- кожний Товар може належати лише одній Категорії;
- кожний Продавець може виставляти на торг один або декілька Товарів;
- кожний Товар може мати лише одного Продавця.

З виділених висловлювань можна виділити наступні зв'язки:

- 1) Покупець здійснює Ставку.
- 2) Ставка здійснюється на Лот.
- 3) Лот складається з Товарів.
- 4) Продавець продає Товар.
- 5) Категорія включає до себе Товари.
- 6) Товар продається при Ставці.

Створені зв'язки відносяться до типу «один-до-багатьох», зовнішні ключи яких посилаються на батьківські сутності і не можуть містити порожні значення. Також необхідно зазначити, що батьківські сутності можуть існувати, навіть якщо на них не посилається жодна з дочірніх. Наприклад, припускається існування покупців, що не опублікували жодного товару. Тому всі зв'язки можуть мати потужність 0 або 1.

Із зроблених припущень створили ER-діаграму, яка наведена на рисунку 1.2.

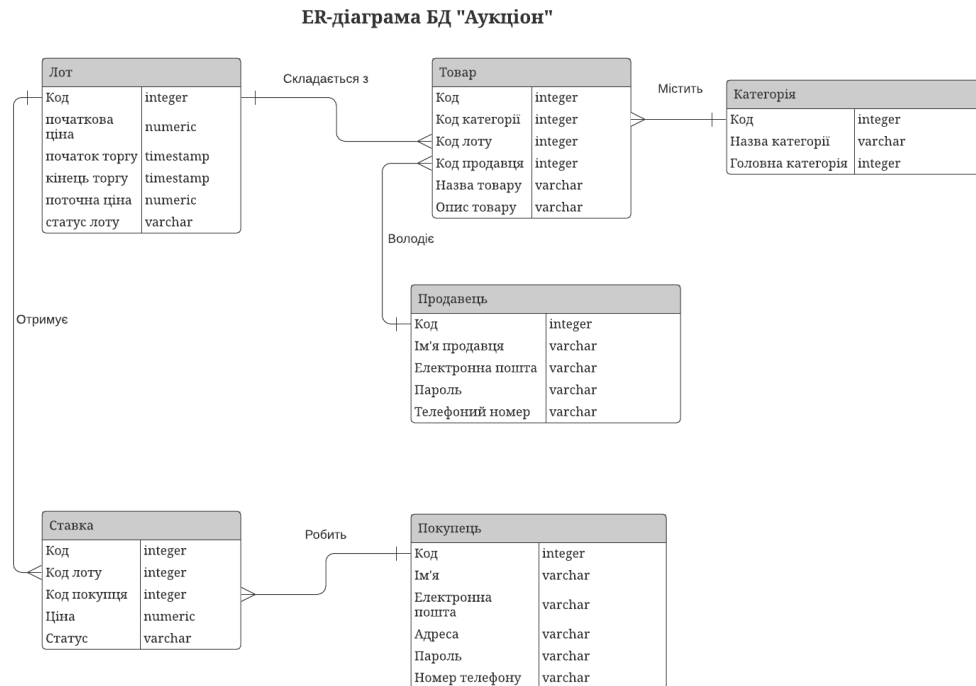


Рисунок 1.2 — ER- діаграма предметної області

1.3 Мета і завдання системи

Системою управління є онлайн-аукціон, метою якого є автоматизація торгів, групування їх за категоріями, ведення обліку проданих лотів, визначення з них актуальних. Головним завданням даної системи є організація ефективної роботи онлайн-аукціону.

1.4 Призначення системи

Системою можуть користуватися: адміністратор, продавець, покупець, а також незареєстрований користувач.

Так, онлайн-аукціон дозволить покупцям робити ставки, виконувати пошук за бажаними критеріями, отримувати повідомлення на поштову скриньку про результати торгів не виходячи з дому.

Основною відмінністю продавців від покупців є те, що вони можуть додавати товари, однак робити ставки вони не можуть. На діаграмах використання системи Use Case (рис.1.3-1.6) продемонстровані основні можливості кожної групи користувачів.

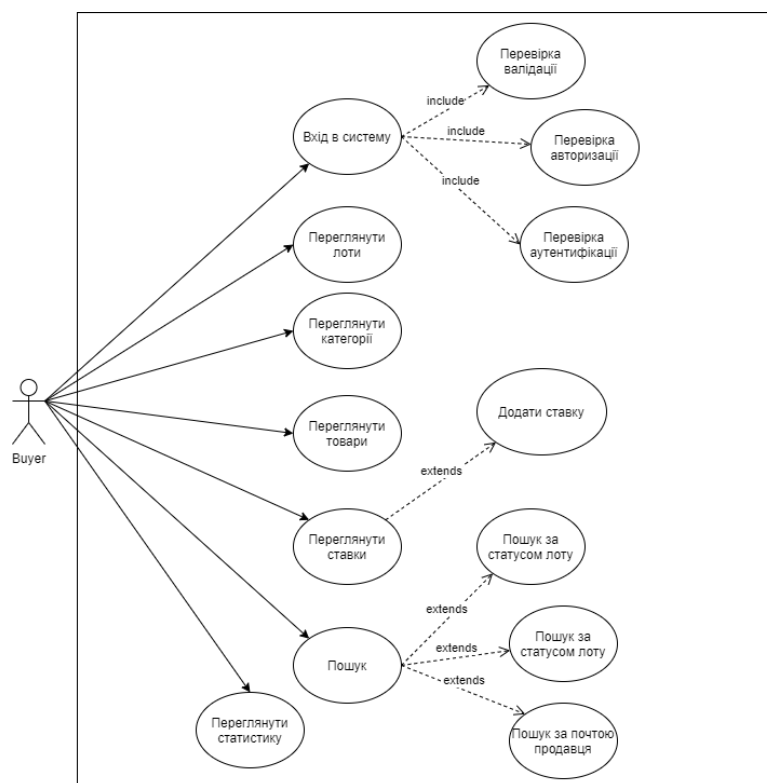


Рисунок 1.3 — Діаграма варіантів використання для покупця

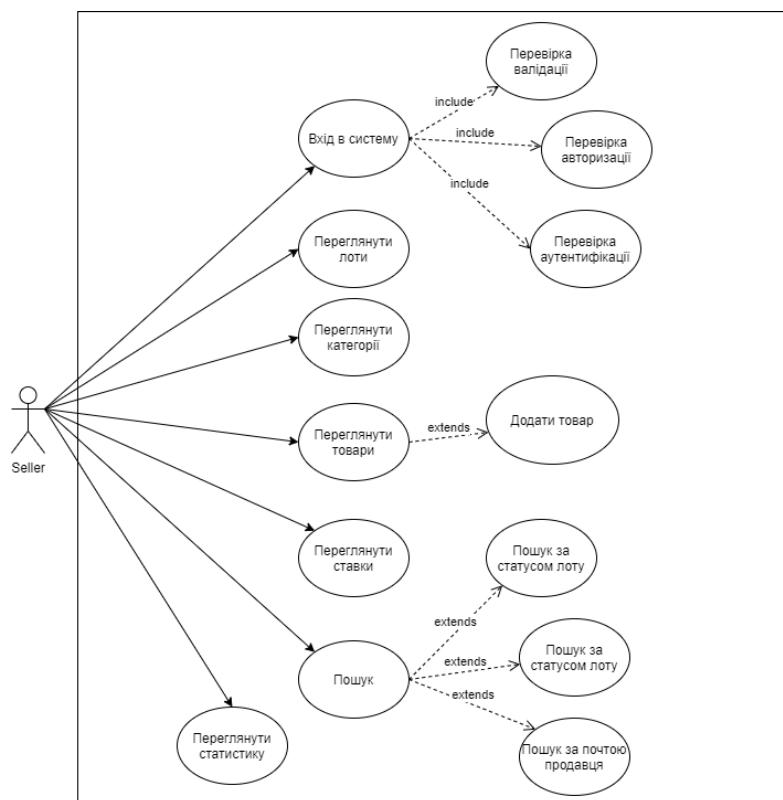


Рисунок 1.4 — Діаграма варіантів використання для продавця

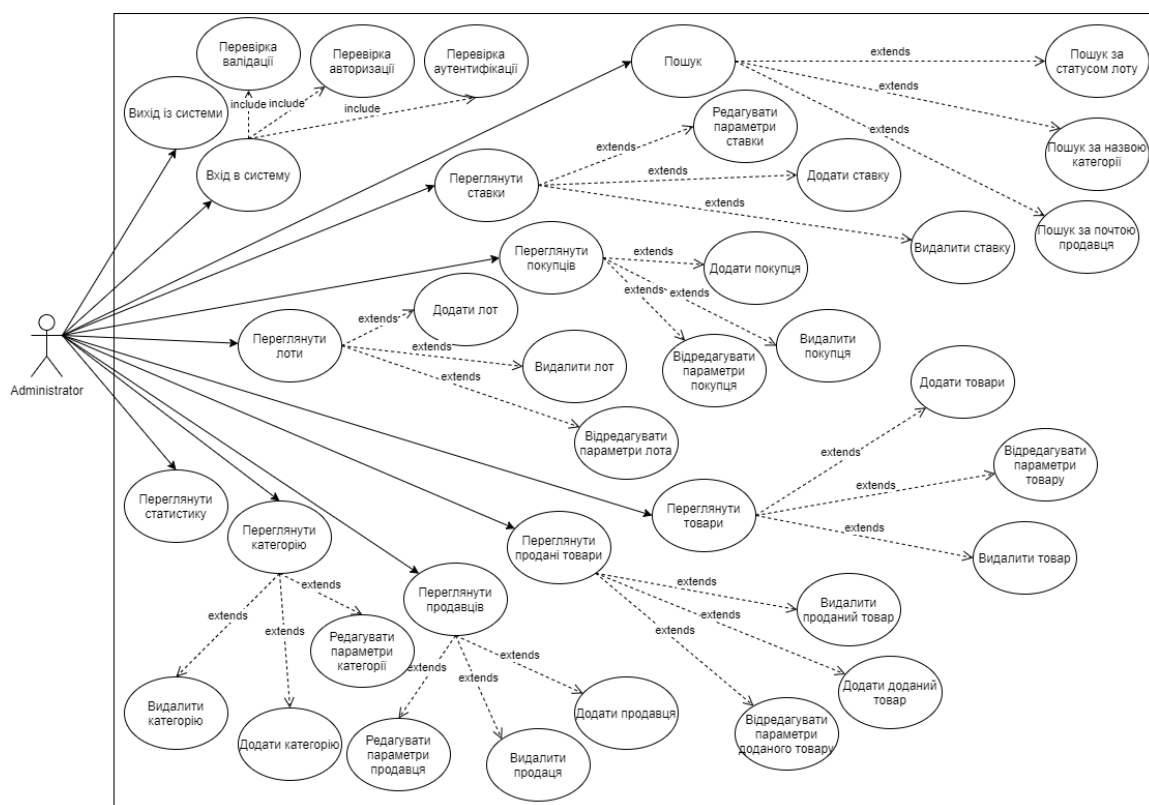


Рисунок 1.5 — Діаграма варіантів використання для адміністратора

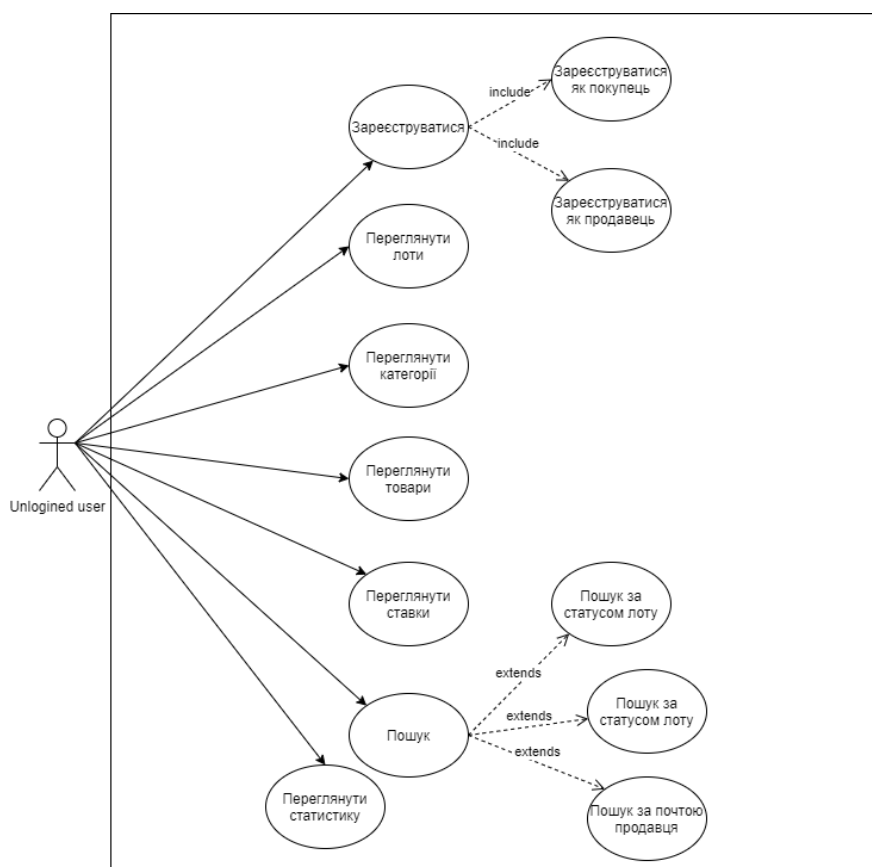


Рисунок 1.6 — Діаграма варіантів використання для незареєстрованого користувача

1.5 Вимоги до системи

Завдання полягає в створенні системи для онлайн-аукціону. Об'єктами такої системи є: товари, лоти, продавці, покупці, ставки, категорії.

Керуючись бізнес-логікою, система повинна виконувати наступні завдання: при додаванні або редагуванні ставки, її ціна не може бути нижчою за попередню, продавець та покупець, який став переможцем торгів повинен отримувати повідомлення на пошту про результати, при видаленні ставок необхідно обрати найбільшу з існуючих, при додаванні або редагуванні лоту дата кінця не може бути меншою за дату початку, при видаленні лоту, система повинна додавати його товари до проданих товарів, при додаванні покупця чи продавця його пошта та номер телефону мають бути унікальними в системі в цілому.

Також система має наступні форми: продані товари, переможці всіх торгів, лоти, які доступні для ставок.

1.6 Висновки до розділу 1

В даному розділі провели аналіз предметної області з метою визначення специфіки подальшої розробки системи. Основними сутностями стали товар, покупець, продавець, категорія, ставка, лот.

Визначили та описали тип та особливості взаємозв'язків між ними, на основі яких побудували ER-діаграму сутностей. Виявили, що всі взаємозв'язки можуть мати потужність 0 або 1.

За допомогою діаграм Use Case розбили функціональні можливості за такими типами користувачів: адміністратор, покупець, продавець, незареєстрований користувач. Основною відмінністю між покупцем та продавцем стала можливість робити ставки (для покупця) та додавати товари (для продавця), незареєстрований користувач позбавлений будь-якою змогою модифікувати дані, що, безумовно, підвищує надійність системи. Адміністратор може виконувати будь-які дії з наявних у системі.

Сформували вимоги до системи та обмеження бізнес-логіки, визначили основні форми, які вимагатимуть реалізації. Перейдемо до опису проектування системи

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Проектування бази даних

При проектуванні бази даних для підвищення надійності буде використовуватись механізм реплікації, що дозволить відновити дані у разі необхідності. Реплікація — механізм, який дозволяє копіювати дані з одного вузла на інший. Для нашої система реплікація матиме форму «master-slave», де буде 1 slave та один master.

2.1.1 Опис таблиць бази даних

Опис таблиць БД на основі аналізу бази даних із вказаними типами полів, обмеженнями цілісності описом наведено у таблиці 2.1.

Таблиця 2.1 — Опис таблиць БД

Назва таблиці	Поля	Опис	Тип даних	Обмеження
buyer(сутність покупець)	id_buyer	Ідентифікаційний код покупця	integer	Первинний ключ
	buyer_name	ПІБ покупця	varchar	
	email_buyer	Електронна пошта покупця	varchar	Повинна мати символи “@” та “.”
	address_buyer	Адреса покупця	varchar	
	password_buyer	Пароль покупця	varchar	
	phone_number_buyer	Телефонний номер покупця	varchar	Має містити 10 символів повинен бути унікальним
lot(сутність лот)	id_lot	Ідентифікаційний код лоту	integer	Первинний ключ
	start_price	Початкова ціна	numeric	
	current_price	Поточна ціна	numeric	

	start_of_auction	Дата початку продажу лоту	timestamp	
	end_of_auction	Дата кінця продажу лоту	timestamp	
	status_lot	Статус лоту	varchar	
bid(сутність ставка)	id_bid	Ідентифікаційний код ставки	integer	Первинний ключ
	id_buyer	Зовнішній ключ на таблицю покупця	integer	Зовнішній ключ, каскадне оновлення, каскадне видалення.
	id_lot	Зовнішній ключ на таблицю лот	integer	Зовнішній ключ, каскадне оновлення, каскадне видалення;
	Price	Ціна ставки	numeric	
	status_bid	Статус ставки	varchar	
category(сутність категорія)	id_category	Ідентифікаційний код категорії	integer	Первинний ключ
	category_name	Назва категорії	varchar	
	patern_category	Головна категорія	integer	
seller(сутність продавець)	id_seller	Ідентифікаційний код продавця	integer	Первинний ключ
	name_seller	Ім'я продавця	varchar	
	email_seller	Електронна пошта продавця	varchar	Повинна мати символи “@” та ”.”
	password_seller	Пароль продавця	varchar	
	phone_number_seller	Телефонний номер продавця	varchar	Має містити 10 символів та номер телефону повинен бути унікальним
item(сутність товар)	id_item	Ідентифікаційний код товару	integer	Первинний ключ

	id_seller	Зовнішній ключ на таблицю продавець	integer	Зовнішній ключ, каскадне оновлення, каскадне видалення
	id_of_category	Зовнішній ключ на таблицю категорія	integer	Зовнішній ключ, каскадне оновлення, видалення set null)
	name_of_item	Назва товару	varchar	
	description	Опис товару	varchar	
	id_lot	Зовнішній ключ на таблицю лот	integer	Зовнішній ключ, каскадне оновлення, каскадне видалення

2.1.2 Проектування логічної схеми БД

На основі отриманого опису таблиць побудували логічну модель бази даних, яка зображена на рисунку 2.1.

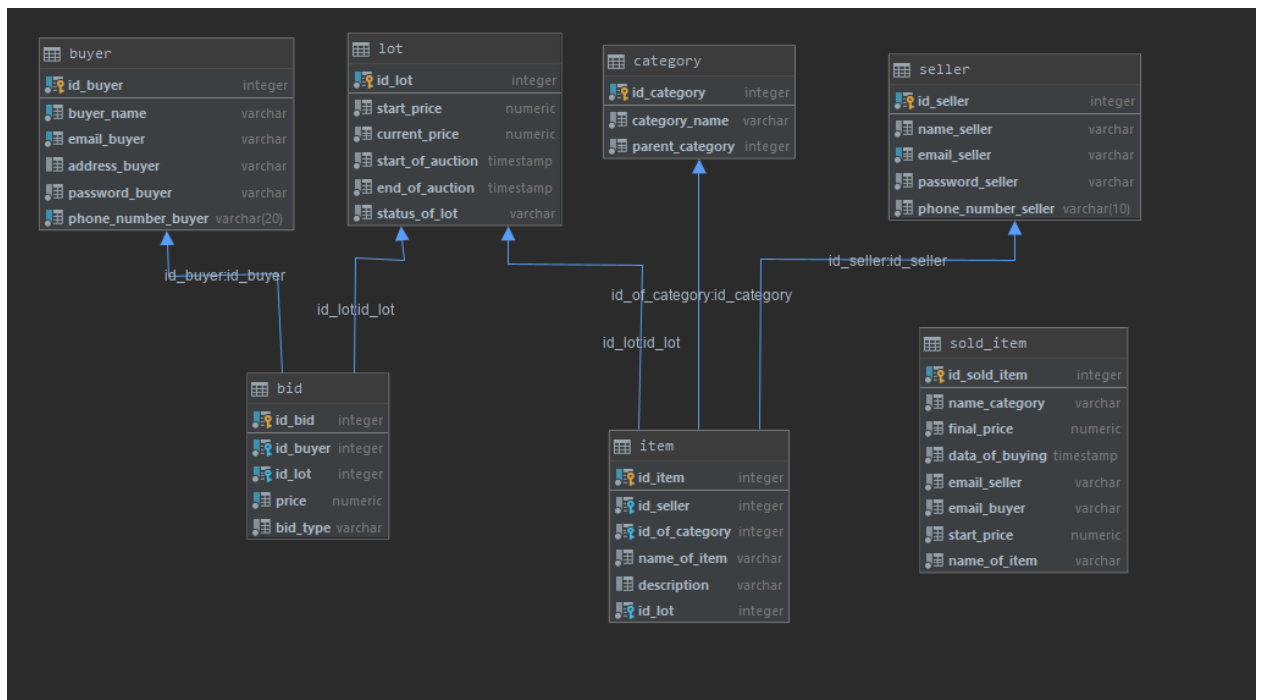


Рисунок 2.1 — Логічна схема бази даних

База даних «Аукціон» має чітку неподільність значень в кожному полі, тобто значення кожного поля атомарне, записи таблиці не впливають один на одного та розміщення записів в таблиці ніяк не впливає на інші, так як кожний із рядків унікальні, а також що є дуже важливим для нормалізації першого рівня, первинний ключ для кожної із створених таблиць не повторюється в межах кожної з них і має конкретну роль ідентифікатора запису.

Варто зазначити, що дана інформаційна система характеризується повною відсутністю залежності неключових полів від частини складного ключа, а також виключається залежність неключових полів один від одного, тобто усунена транзитивність між атрибутами.

Також можна помітити, що база даних не містить багатозначних залежностей між атрибутами, наприклад, таблиці bid та lot пов'язані один з одним, а lot в свою чергу містить зв'язок із сутністю item і вони не впливають один на одного, що підвищує рівень нормалізації до 4НФ.

Коли будь-яка залежність по з'єднанню в ньому визначається тільки його потенційним ключами і залежність по самому з'єднанню представляється ні багатозначною ні функціональною можна сказати, що система матиме 5НФ, що справедливо і для нашої БД, наприклад, кожна таблиця не містить таких полів, які б суперечили один одному або дублювали факти один про одного. Якби інформація про товари, лоти та ставки містилася б в одній таблиці, то довелося б дублювати інформацію про те, що товар належить тому самому лоту при зміні ставки, до того ж додавання товару до лоту змусило б нас дублювати інформацію про ставку на цей лот.

Із вище перелічених характеристик можна зробити логічний висновок, що дана БД представлена у 5НФ.

2.1.3 Проектування бізнес-логіки та бізнес-правил

Для забезпечення коректної роботи бізнес-логіки та бізнес правил було обрано використання тригерів та збережених процедур, зокрема:

Таблиця 2.2 — Опис трігерів

Назва триггеру	Опис	Коли виконується та подія, що викликає
Таблиця bid(ставки)		
del_func	Необхідний для вибору нової актуальної ставки у разі наявності інших ставок на даний лот у випадку видалення поточної найвищої ставки	AFTER DELETE
insert_func	Перевірка ставки, яку намагається зробити покупець на не нульове значення ціни, а також на те щоб додана ставка була більша за попередню актуальну у разі успішної перевірки зміна поточної ціни лоту на ціну доданої ставки	BEFORE INSERT
update_bid_af_func	Зміна статусу всіх інших ставок на Refused, а також зміна поточної ціни лоту на відредаговану	AFTER UPDATE
update_bid_func	Перевірка ставки, яку намагається відредагувати користувач на те, щоб вона була не нижча за поточну, а також не була нулем	BEFORE UPDATE
Таблиця lot(лот)		

deletelot_func	Перенесення даних лоту до таблиці проданих товарів	BEFORE DELETE
update_func	Перевірка на те, щоб дата початку не була пізнішою за дату кінця аукціону, а також перевірка, щоб поточна ціна не була меншою за початкову у разі редагування	BEFORE UPDATE

Виконання бізнес-логіки без додаткових збережених процедур, опис яких наведено у таблиці 2.3.

Таблиця 2.3 — Опис процедур

Назва процедури	Опис	Вхідні параметри	Вихідні параметри
outp	Використовується тригером для видалення з таблиці лот для збору інформації з інших таблиць до таблиці логування solditem відносно даного лоту	Вхідними параметрами функції є: id лоту відносно якого потрібно проводити збір інформації	Вихідними параметрами є таблиця, з наступними колонками: назва категорії(varchar), остаточна ціна(numeric), дата продажу(timestamp), поштова скринька покупця(varcha

			r), поштова скринька продавця(varchar), початкова ціна(numeric), а також назва товару(varchar)
insertsold	Використовується для вставки до таблиці логування solditem даних зібраних таблицею outp	Вхідними параметрами функції є: назва категорії(varchar), остаточна ціна(numeric), дата продажу(timestamp), поштова скринька покупця(varchar), поштова скринька продавця(varchar), початкова ціна(numeric), а також назва	Дана функція нічого не повертає

		товару(varchar)	
clear_close dlots	Використовується для видалення усіх лотів, які мають статус «Closed»	Не приймає жодних параметрів	Повертає набір записів таблиці лот, які стали результатом видалення

Також для логування необхідно створити таблицю, яка буде зберігати продані лоти, опис наведено у таблиці якої наведено у таблиці 2.4.

Таблиця 2.4 — Опис таблиці solditem

Назва таблиці	Поля	Опис	Тип даних	Обмеження
sold_item(сутність проданого товару)	id_sold_item	Ідентифікаційний код проданого товару	integer	Первинний ключ
	name_category	Назва категорії	varchar	
	final_price	Остаточна ціна	double	
	data_of_buying	Дата продажу товару	timestamp	
	id_item	Зовнішній ключ на таблицю товар	integer	Зовнішній ключ, каскадне оновлення, видалення(set null)
	email_seller	Електронна пошта продавця	varchar	Повинна мати символи “@” та “.”
	email_buyer	Електронна пошта покупця	varchar	Повинна мати символи “@” та “.”
	start_price	Початкова ціна	money	
	name_of_item	Назва товару	varchar	

За допомогою діаграм діяльності (рисунки 2.2—2.6) описали процес модифікації таблиць, які працюють з тригерами.

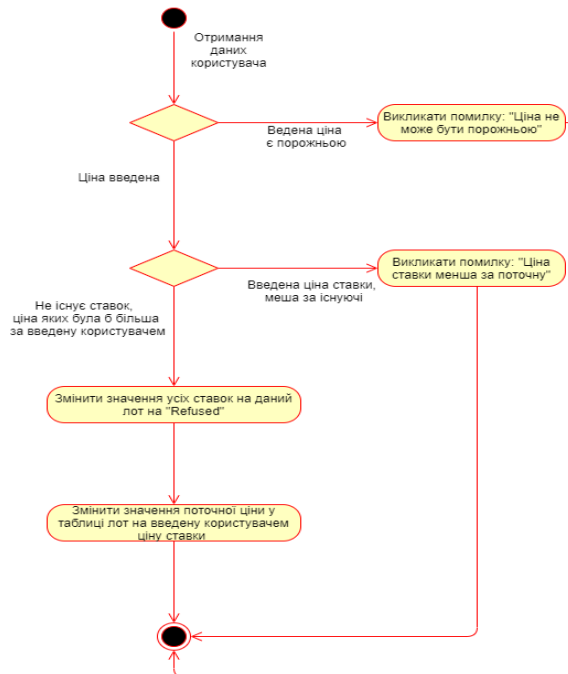


Рисунок 2.2 — Діаграма діяльності процесу додавання рядку до таблиці ставок(bid)

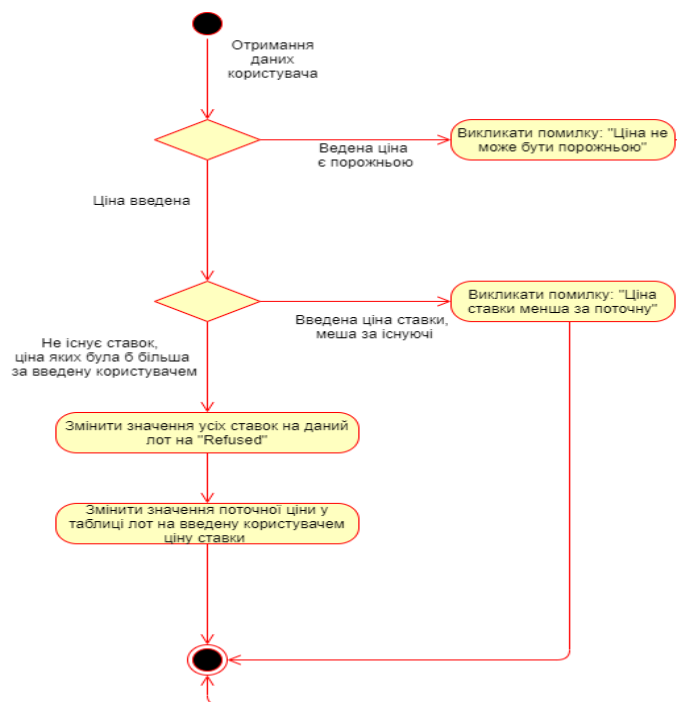


Рисунок 2.3 — Діаграма діяльності процесу редагування таблиці ставок(bid)

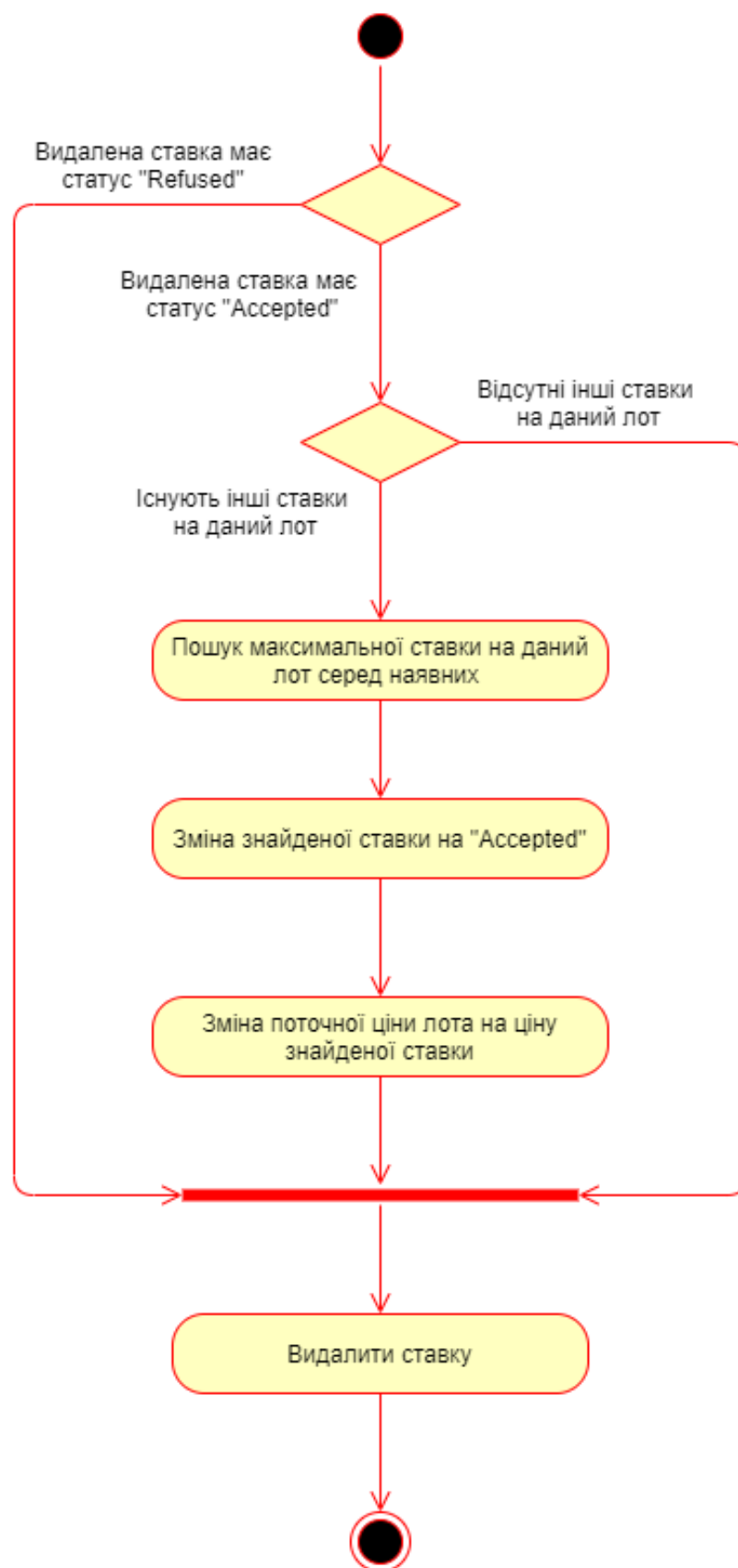


Рисунок 2.4 — Діаграма діяльності процесу видалення із таблиці ставок(bid)

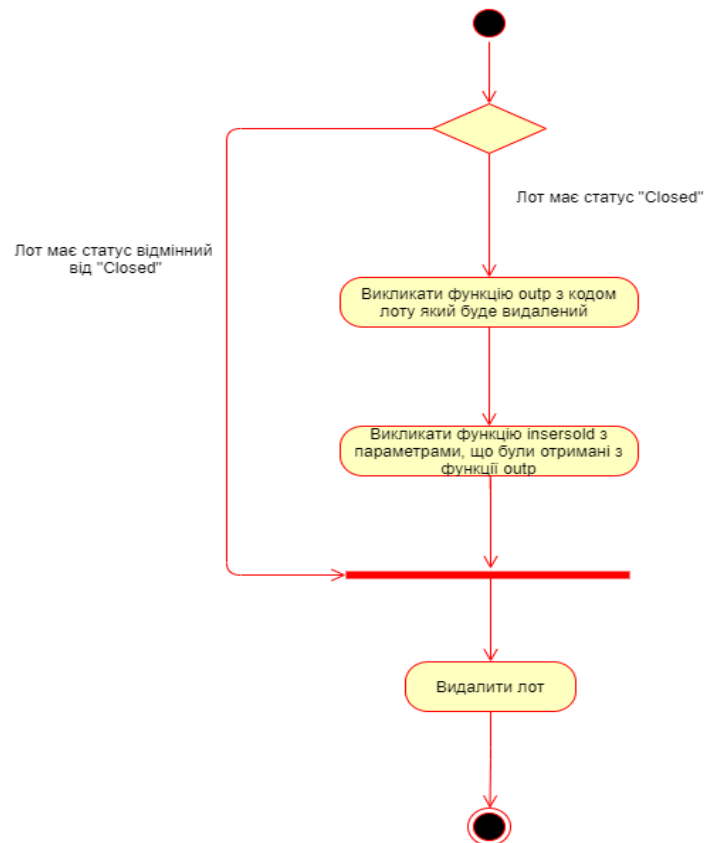


Рисунок 2.5 — Діаграма діяльності процесу видалення із таблиці лотів(lot)

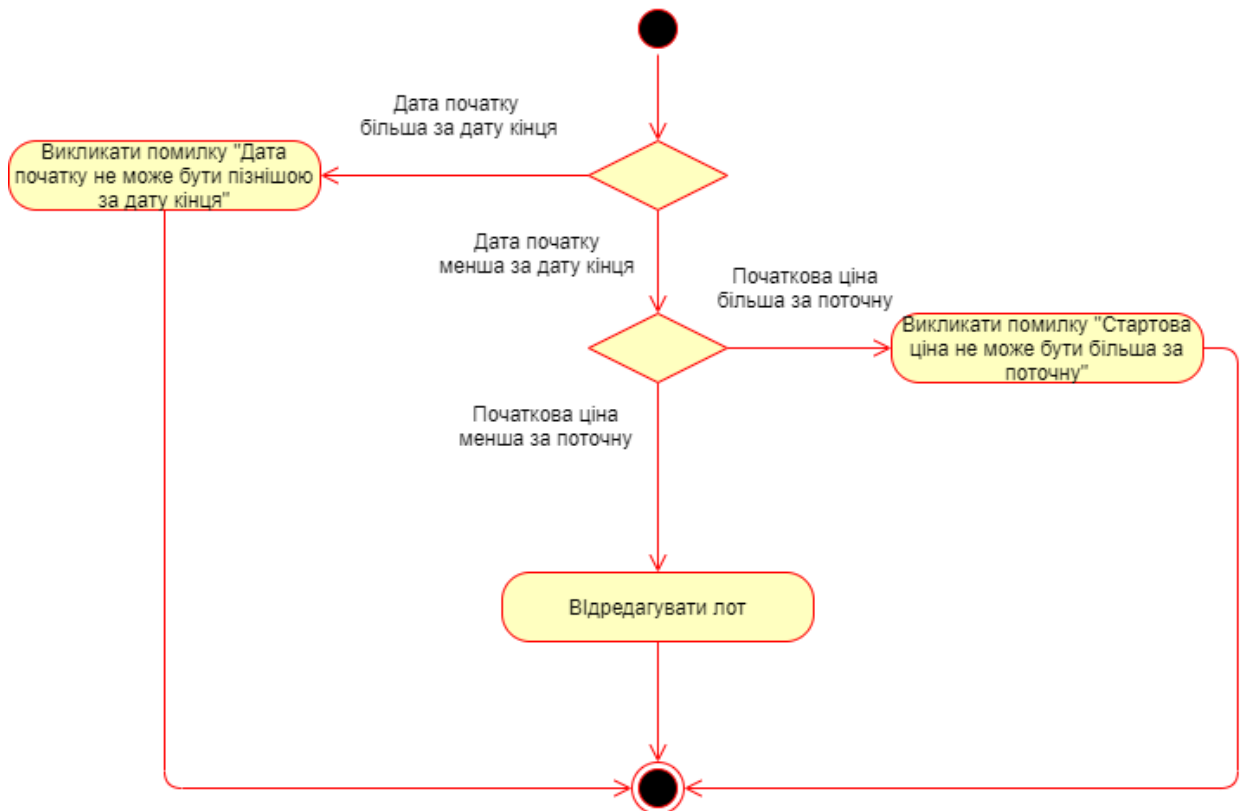


Рисунок 2.6 — Діаграма діяльності процесу редагування таблиці лотів(lot)

2.2 Проектування архітектури системи

Архітектура системи відповідає типу клієнт-сервер, призначена для організації доступу користувачів до бази даних (рисунок 2.7). Її можна розділяти їх на чотири рівні:

- рівень інтерфейсу користувача;
- рівень сервісів;
- рівень обробки даних;
- рівень даних.

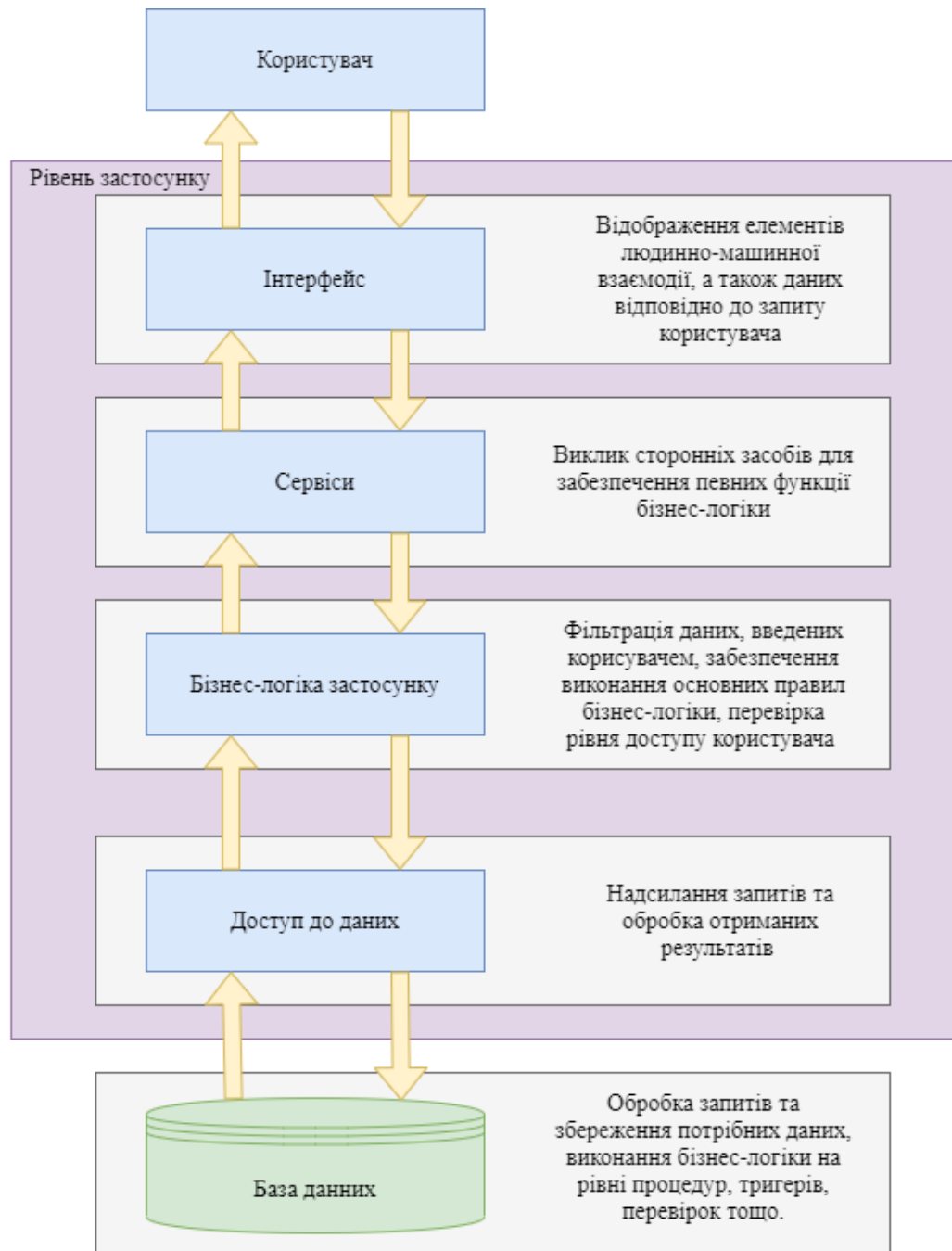


Рисунок 2.7 – Архітектура системи

1) Рівень користувацького інтерфейсу

Даний рівень реалізується на клієнті та містить набір елементів людино-машинної взаємодії, за допомогою яких користувач може надсилати запити до системи та отримувати результати у зрозумілій для більшості формі.

2) Рівень сервісів

Рівень сервісів необхідний для інтеграції сторонніх елементів, які будуть використовуватися у системі. Наприклад, для реалізації можливості надсилання листів до електронної пошти користувачів необхідно забезпечити зв'язок з SMTP-службою.

3) Рівень бізнес-логіки

Даний рівень складається з наступних частин: обробка отриманої інформації від сервісу, перевірка валідності даних, підготовка їх до роботи з базою, реалізація основної функціональності системи. На противагу користувацьким інтерфейсам, сервісам або базам даних на даному рівні відбувається визначення загальних закономірностей.

4) Рівень доступу до даних

Рівень даних у моделі клієнт-сервер містить програми, які надають дані системам, що їх оброблюють. Основною властивістю даного рівня є необхідність підтримки зв'язку з базою на рівні запитів, конвертація даних інших рівнів у зрозумілий для БД формат та навпаки.

5) Рівень бази даних

Специфічною властивістю цього рівня є вимога живучості та можливість самовідновлення. Отже, дані повинні зберігатися, розраховуючи на подальше використання. Використання файлової системи для даного функціоналу є ресурсномістким варіантом, а також даний формат значно знизить надійність, тому використаємо СКБД та створимо базу даних.

2.3 Вибір інструментальних засобів розробки системи

2.3.1 Сервер баз даних

На сьогоднішній день існує велика кількість серверів для роботи з базою даних, кожна з яких має власні переваги та недоліки. Одними з

найпопулярніших є MongoDB, PostgreSQL, MySQL. Тому створили порівняльну таблицю (таблиця 2.5) з метою визначення серверу баз даних для роботи та зваживши важливі для нас переваги та недоліки обрали PostgreSQL.

Таблиця 2.5 — Порівняння різних СКБД.

Критерій відносно системи	PostgreSQL	MySQL	MongoDB
Опис	Об'єктно-реляційна база даних, яка працює за однією методикою storage engine. Таблиці представлені у вигляді об'єктів, вони можуть успадковуватися, а всі дії виконуються за допомогою об'єктно-орієнтованих функцій[1]	Реляційна СКБД, яка використовує різні методики для збереження даних у таблицях, однак робота з ними «схована», на синтаксис методика не впливає. Різниця між методиками полягає у зчитуванні та запису даних. [1]	Документно-орієнтована система управління базами даних, яка не потребує опису схеми таблиць. Вважається одним з класичних прикладів NoSQL-систем, використовує JSON-подібні документи і схему бази даних.[2]
Переваги	1. Зручна кластеризація 2. Наявність досвіду роботи 3. Гнучка структура схеми даних	1. Ефективне керування структурними даними невеликого розміру	1. Гнучка до зміни складу даних 2. Ефективна для збереження високих обсягів даних

	<p>4. Зручне розмежування доступу</p> <p>5. Ефективна оптимізація та кешування</p>	2. Зручна реплікація	<p>3. Зручна у налаштуванні</p> <p>4. Простота реплікації</p>
Недоліки	<p>1. Складна реплікація</p> <p>2. Відсутність авторизації за замовчуванням</p> <p>3. Складність зміни тригерних функцій та збережених процедур(необхідність видалення)</p>	<p>1. Чутливість до нестабільності серверу</p> <p>2. Трудомісткий процес редагування структури бази</p> <p>3. Складність кластеризації</p>	<p>1. Відсутність простих транзакцій</p> <p>2. Низька зв'язність даних</p> <p>3. Складність реплікації</p>
Висновок	Використання у нашій системі дозволить зекономити час через наявність досвіду налаштування та роботи, також висока оптимізація є значною перевагою, недоліки даної СКБД можна компенсувати	Трудомістке редагування структури та відсутність гнучкості відносно інших СКБД значно ускладнить роботу, переваги ж відносно	Відсутність високої зв'язності між об'єктами є вагомим недоліком, який значно ускладнить роботу з СКБД.

	наявністю великої кількості документації.	PostgreSQL є незначними.	
--	---	--------------------------	--

2.3.2 Технології реалізації системи

Кожний шар архітектури має власні технології та мови програмування, які він використовує.

Технології рівня інтерфейсу

Для розробки інтерфейсу була обрана технологія Java Server Pages, яка дозволяє розроблювати web-сторінки, що містять як статичні, так і динамічні компоненти. Текст статичної частини таких сторінок оформлений у наступних форматах:

HTML (Hyper Text Markup Language) — формат, який використовується для розмітки сторінки. Включає до свого складу простий набір команд, за допомогою яких можна зручно описувати структуру документу та виділяти логічні одиниці (абзаци, блоки елементів, кнопки, тощо). Також за допомогою цієї мови виконані посилання між іншими документами.

CSS (Cascading Style Sheets)[5] — формат, який використовується для опису зовнішнього вигляду елементів, що були «намічені» за допомогою HTML.

Також для дизайну нашої системи був використаний Bootstrap — фреймворк, який містить у собі широкий набір інструментів для web-дизайну, готові шаблони оформлення за допомогою HTML та CSS елементів.

Засобами для розробки динамічної частини сторінки стали :

EL (Expression Language) — скриптова мова виразів, що дозволяє отримати доступ всередині сторінки до Java-компонентів. В нашому проекті за допомогою цієї мови виконується передача даних з серверу додатку до сторінки.

JSTL (JSP Standard Tag Library) — розширення стандартної специфікації JSP для можливості спрощення коду сторінки та використання звичних для нас умов, циклів, порівнянь, тощо.

JavaScript — мова кодування, яка використовується для наповнення наших сторінок динамічними елементами такими як графіки, вікна повідомлень тощо.

Технології рівня сервісу

Сервісна частина нашої системи використовує наступні технології:

JavaMail — API, який призначений для роботи з електронними поштами, використовуючи мережеві протоколи різного формату (наша система використовує SMTP), призначений для передачі листів до електронної пошти. Ця служба використовується для сповіщення продавців та покупців про результати торгів.

Google Visualization API — web-сервіс, який дозволяє створювати графічні діаграми та налаштовувати їх відносно даних у системі. Працює за принципом відправки запиту до Google-сервісу з даними та налаштуваннями майбутньої діаграми, після чого сервер Google створює PNG-зображення за даними з запиту користувача. Використовується для відображення графіків на сторінці статистики за запитом користувача.

Технології рівня бізнес-логіки

Java Servlet API[4] — стандартизований Java API, який необхідний для роботи з сервером за принципом запит-відповідь. Після того як користувач обирає певну дію на JSP-сторінці вона надсилає запит до сервлета з параметрами. З запиту він отримує дані, оброблює їх та повертає результат на сторінку у вигляді відповіді. Саме у них і реалізована бізнес-логіка додатку.

Технології рівня доступу до даних

Для доступу до даних використовується технологія ORM (Object-Relational-Mapping) — технологія програмування, яка пов'язує базу даних із об'єктом який використовується для об'єктно-орієнтованих мов програмування, створюючи об'єктну базу даних. У випадку Java такими

об'єктами є класи. Така технологія підвищує зручність роботи із системою, а також підвищує безпеку, оскільки робота з даними відбувається безпосередньо. В нашому проекті для реалізації даної технології було використано специфікацію JPA (Java Persistence API), бібліотека якої — Hibernate є найпопулярнішою для побудови даної технології.

2.4 Проектування модулів системи

2.4.1 Проектування модулю доступу до даних

Оскільки для реалізації доступу до даних була обрана технологія ORM, побудували діаграму класів сутностей (рисунок 2.8).

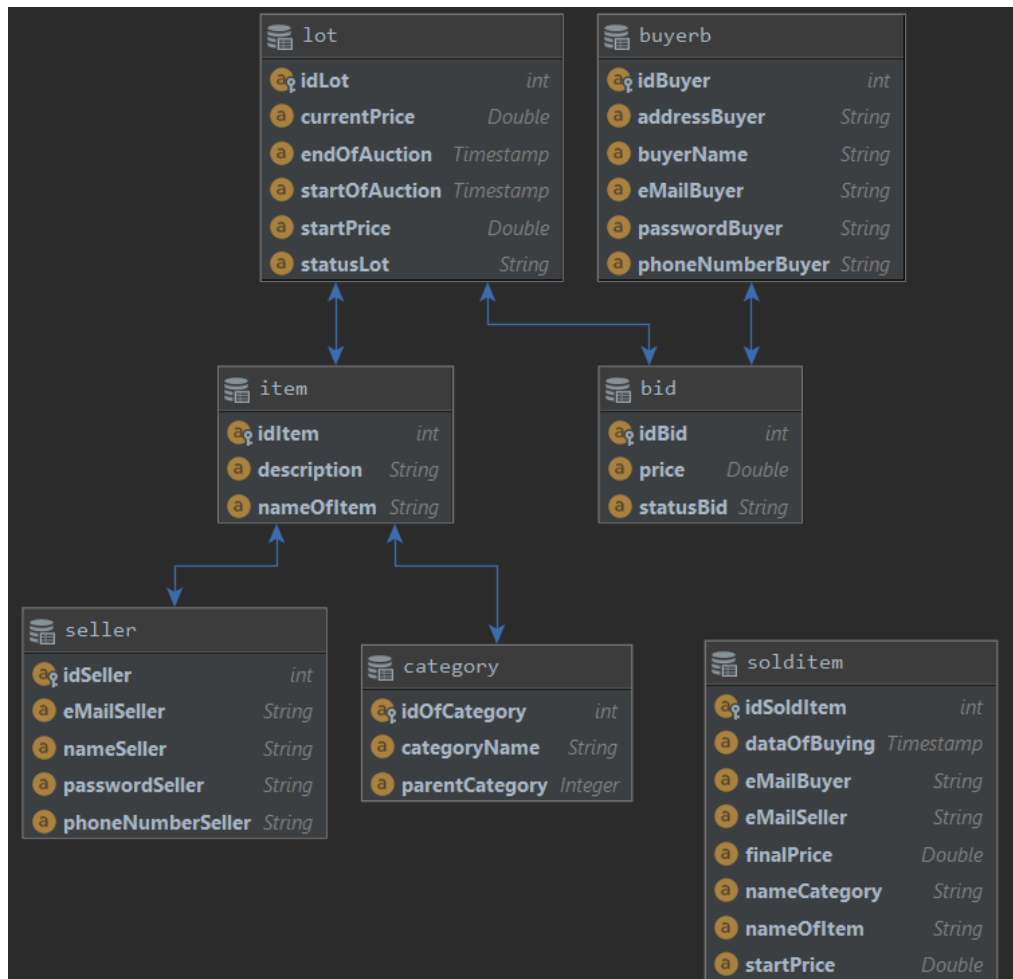


Рисунок 2.8 — ERD-діаграма класів

З діаграми можна побачити, що відповідно до визначених у розділі 1 сутностей було згенеровано класи, значення полів яких наведено у таблиці 2.6. Зв'язки у такій реалізації представлено у вигляді посилань на об'єкти типу сутності, з якою вони мають бути.

Таблиця 2.6 — Опис класів сутностей БД

Назва класу	Поля класу	Поля у БД	Опис	Тип даних класу
buyerb (сутність покупець)	idBuyer	id_buyer	Ідентифікаційний код покупця	integer
	buyerName	buyer_name	ПІБ покупця	String
	eMailBuyer	email_buyer	Електронна пошта покупця	String
	addressBuyer	address_buyer	Адреса покупця	String
	passwordBuyer	password_buyer	Пароль покупця	String
	phoneNumberBuyer	phone_number_buyer	Телефонний номер покупця	String
lot(сутність лот)	idLot	id_lot	Ідентифікаційний код лоту	integer
	startPrice	start_price	Початкова ціна	numeric
	currentPrice	current_price	Поточна ціна	numeric
	startOfAuction	start_of_auction	Дата початку продажу лоту	timestamp
	endOfAuction	end_of_auction	Дата кінця продажу лоту	timestamp
	statusLot	status_lot	Статус лоту	String
bid(сутність ставка)	idBid	id_bid	Ідентифікаційний код ставки	integer
	Екземпляр класу сутності покупця (buyerb)	id_buyer	Зовнішній ключ на таблицю покупця	buyerb
	Екземпляр класу сутності покупця (lot)	id_lot	Зовнішній ключ на таблицю лот	lot
	price	Price	Ціна ставки	numeric
	statusBid	status_bid	Статус ставки	bidtype
category(сутність категорія)	idOfCategory	id_category	Ідентифікаційний код категорії	integer
	categoryName	category_name	Назва категорії	String

	parentCategory	parent_category	Головна категорія	integer
seller(сутність продавець)	idSeller	id_seller	Ідентифікаційний код продавця	integer
	nameSeller	name_seller	Ім'я продавця	String
	eMailSeller	email_seller	Електронна пошта продавця	String
	passwordSeller	password_seller	Пароль продавця	String
	phoneNumberSeller	phone_number_seller	Телефонний номер продавця	String
item(сутність товару)	idItem	id_item	Ідентифікаційний код товару	integer
	Екземпляр класу сутності продавця (seller)	id_seller	Зовнішній ключ на таблицю продавець	seller
	Екземпляр класу сутності категорії (category)	id_of_category	Зовнішній ключ на таблицю категорія	category
	nameOfItem	name_of_item	Назва товару	String
	description	description	Опис товару	String
	Екземпляр класу сутності лоту (lot)	id_lot	Зовнішній ключ на таблицю лот	lot

Для фізичного з'єднання з БД на рівні ORM використовується сесія. Для технології Hibernate реалізується за допомогою інтерфейсу `org.hibernate.Interface Session` — даний інтерфейс є основним для виконання між додатком Java та Hibernate, основною функцією сесії є забезпечення виконання операцій читання, створення, оновлення, а також видалення для екземплярів класів сутностей. Кожна операція виконується за допомогою одиниці роботи з БД — транзакцією. Для роботи з об'єктами типу `Transaction` використовуються такі

методи як `begin()` для створення, `commit()` для виконання та `rollback()` для повернення БД в стан до початку виконання транзакції.

2.4.2 Проектування модулю бізнес-логіки і бізнес-правил

За обробку бізнес-логіки та бізнес-правил на рівні програмного забезпечення, а також підтримку зв'язку з бізнес-логікою, реалізованою за допомогою тригерів та збережених процедур та переданою за допомогою рівня доступу до даних, відповідають сервлети, створені відносно кожної сутності.

Навели за допомогою діаграми діяльності принцип роботи кожного з них (рисунок 2.9).

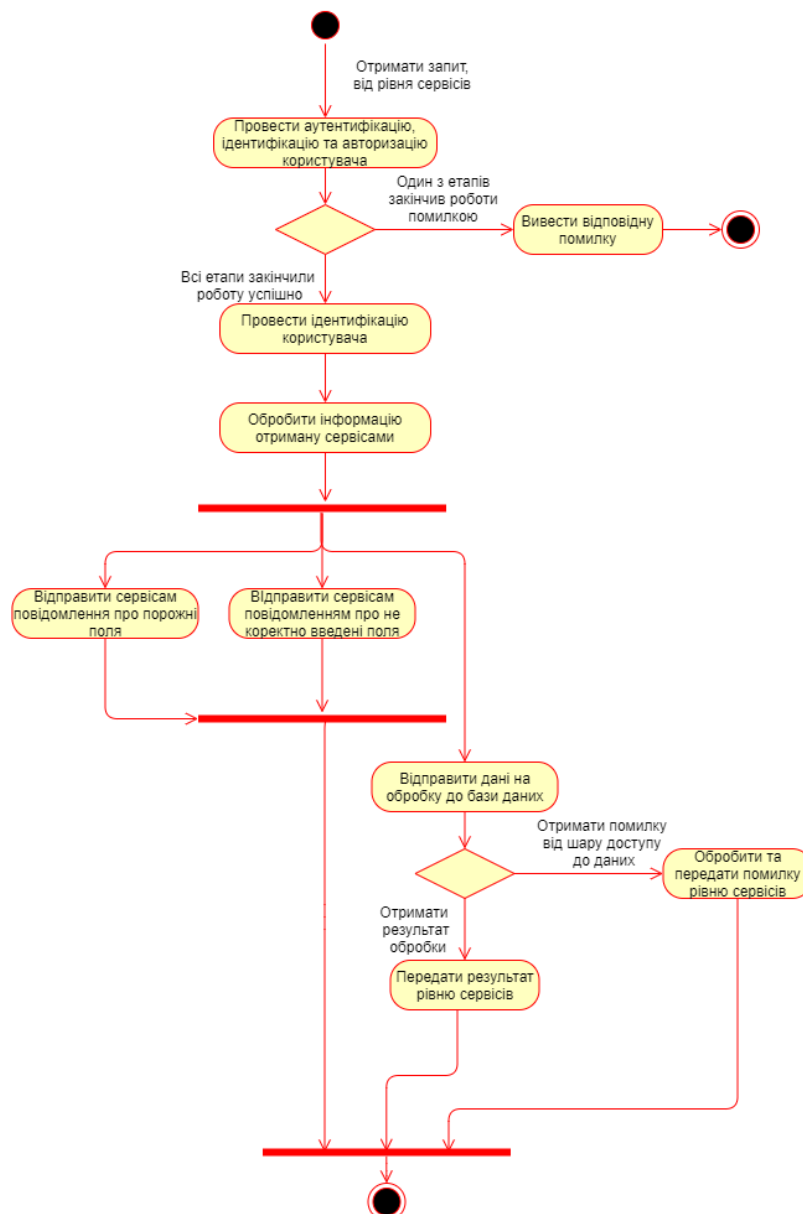


Рисунок 2.9 — Діаграма діяльності

2.4.3 Проектування шару сервісів

Для використання сервісу роботи з поштою був створений клас SendMail, діаграма взаємодії якого з іншими наведена на рисунку 2.10.

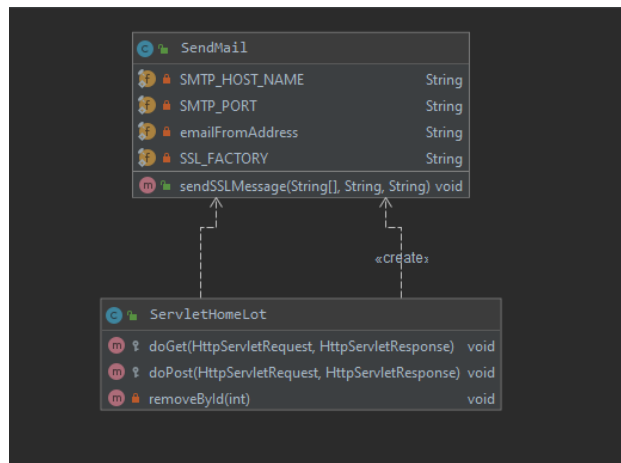


Рисунок 2.10 — Діаграма класів для сервісу обробки та відправки ПОШТИ

Як можна побачити, основними полями класу пошти стали назва серверу, за допомогою якого відбувається відправка повідомлень (SMTP_HOST_NAME) типу String, порт, по якому відбувається відправка (SMTP_PORT) типу String, назва поштової скриньки, з якої відбувається відправка повідомлень (emailFromAddress) типу String, сокет, який використовує система (SSL_Factory). Даний сервіс має метод sendSSLMessage(String[], String, String), де перший параметр — масив поштових скриньок користувачів, які повинні отримати повідомлення, другий — заголовок повідомлення, а третій — текст, який необхідно відправити.

2.4.4 Проектування шару відображення

Створили макет сторінки для виводу даних(рисунок 2.11).

Online Auction

Buyer

Bid

Lots

Items

Category

Sold item

Seller

Рисунок 2.11 — Макет сторінки відображення даних

Кожна сторінка системи має дошку для орієнтації, елементи якої позначені кодом A1–A3. Блок A1 містить посилання на веб-сторінки відображення даних кожної таблиці відповідно до назви. Посилання на сторінку статистики, яка є головною у системі відбувається натиском на «Online Auction». За допомогою блоку A2 можна виконати пошук товару за поштовою скринькою продавця, цей пошук доступний з кожної сторінки системи. Блок A3 дозволяє виконати авторизацію або вийти з облікового запису системи.

Елемент B1 використовується для перегляду та модифікації даних таблиці. Кожний рядок виведеної таблиці відповідає рядку даних у БД. Елементи з кодом B2 та B3 містять у собі посилання на видалення (B2) або сторінку редагування (B3) певного рядку даних.

Для того, щоб додати новий рядок до БД необхідно натиснути на посилання C1, яке перемістить на форму для введення даних.

Також кожна сторінка містить заголовок (елемент D1) для спрощення орієнтації.

Інші пошуки системи також доступні з дошки навігації та «сховані» відносно своєї таблиці. При натиску на які на сторінці відображення виводяться результати. Макети списків для пошуку наведено на рисунках 2.12 —2.13.

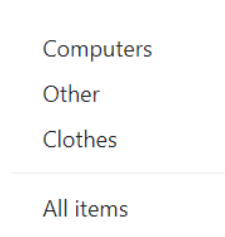


Рисунок 2.12 — Макет списку для пошуку товару за категорією

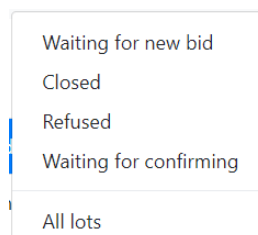


Рисунок 2.13 — Макет списку для пошуку лоту за статусом

Створили макет сторінки для додавання, який наведено на рисунку 2.14.

Рисунок 2.14 — Макет сторінки для додавання

Кожна сторінка для додавання також міститиме навігаційну дошку (елемент А), після чого будуть розташовані елементи для вводу інформації (блок В), для того, щоб зробити інтерфейс інтуїтивно зрозумілим додали надписи для орієнтації. Для відправки даних для перевірки та можливого додавання необхідно натиснути кнопку add (елемент С).

У разі помилки в обробці даних над блоком В виводиться повідомлення про помилку, як це показано на рисунку 2.15.

Рисунок 2.15 — Виведення повідомлення про помилку

Для редагування буде викликатися аналогічна до додавання форма, головною відмінністю стане якої стане заповнений початковими даними блок В.

Оскільки існує можливість авторизуватися, то розробили макет відповідної сторінки (рисунок 2.16).

Рисунок 2.16 —Макет форми для авторизації

Дана сторінка матиме навігаційну панель (блок А), а також поля для вводу поштової скриньки та паролю (блок В), після чого за допомогою кнопки log (блок С) відбувається авторизація, якщо користувач не був зареєстрований, він може це зробити відповідно до бажаних дій (блок D). Певний блок викличе форму для додавання або покупця або продавця залежно від вибору. Після процедури авторизації буде виведено вікно в залежності від ролі користувача. Приклади вікон наведені на рисунку 2.17 —2.19.

Рисунок 2.17 — Повідомлення успішної авторизації для адміністратора

Рисунок 2.18 — Повідомлення успішної авторизації для продавця

Рисунок 2.19 — Повідомлення успішної авторизації для покупця

Початковою сторінкою є вікно статистики, макет якої наведено на рисунку 2.20.

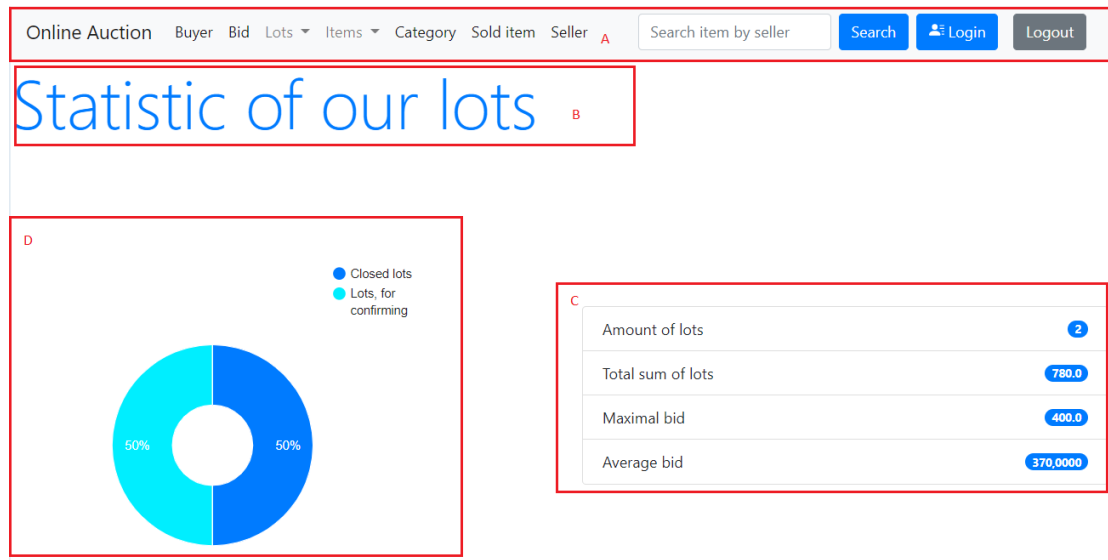


Рисунок 2.20 — Макет сторінки статистики

Як і кожна сторінка сайту, сторінка статистики містить дошку навігації (блок А), після чого знаходиться заголовок до певного фрагменту статистики, що спрощує розуміння інтерфейсу для користувачів(елемент В). Блок С відображує список актуальних на поточний момент параметрів у вигляді списку. Також, деяка статистика відображена за допомогою діаграм(блок D).

2.5 Висновки до розділу 2

На основі проведеного аналізу в розділі був описаний процес проектування системи, в якому визначені основні інструменти та технології, що будуть використані у розробці. Порівняльний аналіз СКБД визначив, що для даної системи найкращим вибором буде саме PostgreSQL через його гнучкість до змін. Виявили специфіку взаємодії між рівнями та інструменти для кожного з них. Спроекували макет майбутньої системи.Перейдемо до опису її розробки.

2 РОЗРОБКА СИСТЕМИ

3.1 Розробка бази даних системи

3.1.1 Розробка фізичної схеми бази даних

На основі логічної схеми бази даних побудували фізичну модель бази даних (рисунок 3.1).

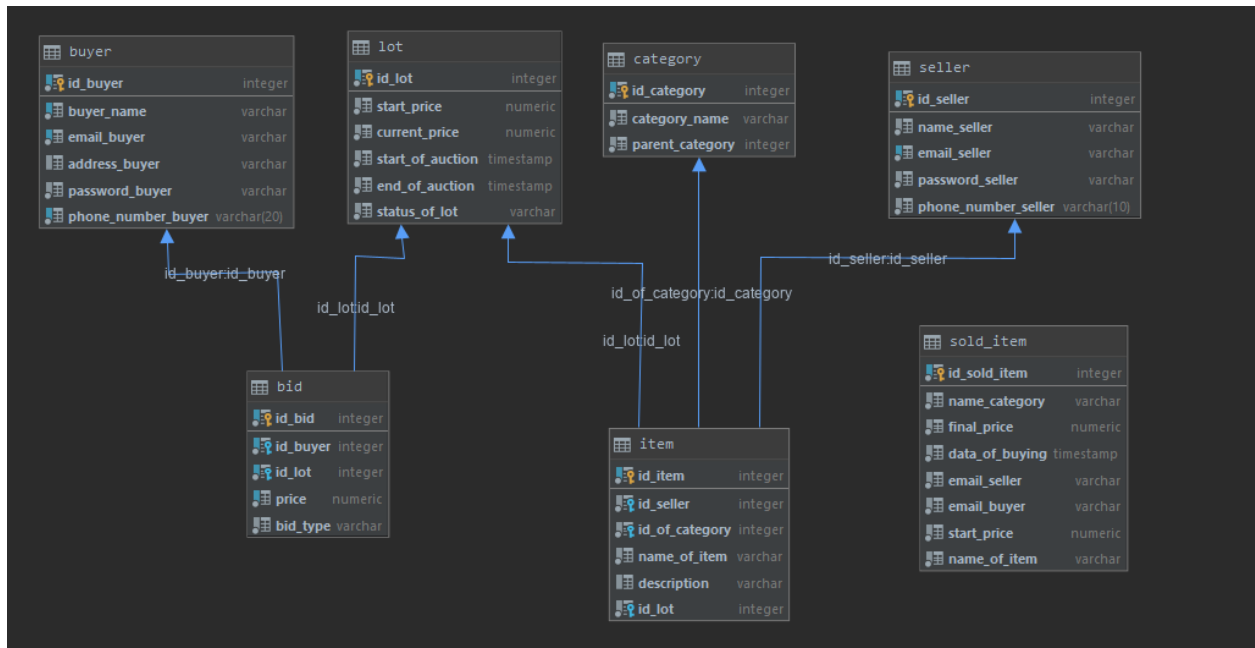


Рисунок 3.1 —Фізична схема бази даних

Перейдемо до детального опису бази даних, створеної в СУБД PostgreSQL.

3.1.2 Забезпечення цілісності даних

Обмеження цілісності створених таблиць наведені на рисунках 3.2—3.8.

Columns						
	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
<input checked="" type="checkbox"/>	id_buyer	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	id_lot	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	price	numeric			<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	bid_type	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	id_bid	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Рисунок 3.2 – Обмеження цілісності таблиці bid

buyer

GeneralColumnsAdvancedConstraintsParametersSecuritySQL

Inherited from table(s)Select to inherit from...

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
	id_buyer	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	buyer_name	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	email_buyer	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	address_buyer	character varying			<input type="checkbox"/>	<input type="checkbox"/>
	password_buyer	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	phone_number_buyer	character varying	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>

Рисунок 3.3 — Обмеження цілісності таблиці buyer

item

GeneralColumnsAdvancedConstraintsParametersSecuritySQL

Inherited from table(s)Select to inherit from...

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
	id_seller	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	id_of_category	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	name_of_item	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	description	character varying			<input type="checkbox"/>	<input type="checkbox"/>
	id_lot	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	id_item	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Рисунок 3.4 — Обмеження цілісності таблиці item

lot

GeneralColumnsAdvancedConstraintsParametersSecuritySQL

Inherited from table(s)Select to inherit from...

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
	id_lot	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	start_price	numeric			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	current_price	numeric			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	start_of_auction	timestamp without time zone			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	end_of_auction	timestamp without time zone			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	status_of_lot	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>

Рисунок 3.5 — Обмеження цілісності таблиці lot

category

GeneralColumnsAdvancedConstraintsParametersSecuritySQL

Inherited from table(s)

Select to inherit from...

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
	category_name	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	parent_category	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	id_category	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Cancel

Reset

Save

Рисунок 3.6 — Обмеження цілісності таблиці category

seller

GeneralColumnsAdvancedConstraintsParametersSecuritySQL

Inherited from table(s)

Select to inherit from...

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
	id_seller	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	name_seller	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	email_seller	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	password_seller	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	phone_number_seller	character varying	10		<input checked="" type="checkbox"/>	<input type="checkbox"/>

Рисунок 3.7 — Обмеження цілісності таблиці seller

sold_item

GeneralColumnsAdvancedConstraintsParametersSecuritySQL

Inherited from table(s)

Select to inherit from...

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
	id_sold_item	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	name_category	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	final_price	numeric			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	data_of_buying	timestamp without time zone			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	email_seller	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	email_buyer	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	start_price	numeric			<input checked="" type="checkbox"/>	<input type="checkbox"/>
	name_of_item	character varying			<input checked="" type="checkbox"/>	<input type="checkbox"/>

Рисунок 3.8 — Обмеження цілісності таблиці solditem

3.1.3 Реалізація бізнес-логіки зі сторони серверу

Реалізували спроектовані у пункті 2.3.1 тригери та збережені процедури, опис яких наведено у таблицях 2.2—2.3.

Створили тригер на додавання , який контролює коли нова ставка на цей лот менша за поточну та коли ціна не заповнена то видає помилку, а також коли нова ставка більша за поточну, то оновлює значення поточної ціни для лоту і змінює стан старої на «Відхилено». Код тригерної функції для додавання рядку до таблиці ставок наведено у лістингу 3.1.

Лістинг 3.1 — запит для створення тригерної функції та тригеру для додавання рядку до таблиці ставок.

```
CREATE FUNCTION insert_func() RETURNS trigger AS
$insert_func$
    BEGIN
        IF EXISTS( SELECT *
                    FROM bid b
                    WHERE b.price>NEW.price AND
b.id_lot=NEW.id_lot)
            THEN
                RAISE EXCEPTION 'price не може бути нижче ніж
поточна!';
            END IF;
        IF NEW.price IS NULL THEN
            RAISE EXCEPTION ' price не може бути 0 ! ';
        END IF;
        IF EXISTS( SELECT *
                    FROM bid b
                    WHERE b.price<NEW.price AND
b.id_lot=NEW.id_lot)
            THEN UPDATE bid SET bid_type='Refused'
WHERE bid.price<NEW.price AND bid.id_lot=NEW.id_lot;
                UPDATE lot SET
current_price=NEW.price WHERE NEW.id_lot=lot.id_lot;
            END IF;
        UPDATE lot SET current_price = NEW.price WHERE
NEW.id_lot=lot.id_lot;
        NEW.bid_type:='Accepted';
        RETURN NEW;
    END;
$insert_func$ LANGUAGE plpgsql;
```

Створили відповідний тригер, який буде використовувати наведену вище функцію, код для створення наведено у лістингу 3.2. Даний тригер і включає в себе три виняткові ситуації, що порушують бізнес-логіку БД, блок обернений у оператор Begin та End і в результаті виконання чи не виконання операції тригер повертає значення NEW. Також для цього тригеру задана умова, що він буде виконуватися до оновлення елемента в таблиці ставок.

Лістинг 3.2 — Запит на створення тригеру на додавання до таблиці ставок

```
$insert_func$ LANGUAGE plpgsql;
CREATE TRIGGER insert_func BEFORE INSERT ON bid
    FOR EACH ROW EXECUTE PROCEDURE insert_func();
```

Редагування таблиці ставок було здійснено за допомогою двох тригерних функцій, одна з яких працює в якості перевірки введених даних, а інша — після того, як значення були змінені, вона відповідає за заміну поточної ціни лоту та стану інших ставок. Код тригерної функції обробки до внесення змін у таблицю ставок наведено у лістингу 3.3

Лістинг 3.3 — Запит на створення тригерної функції на зміну значень після модифікації таблиці ставок.

```
CREATE FUNCTION update_bid_func () RETURNS trigger AS
$update_bid_func $
BEGIN
    IF OLD.price > NEW.price AND OLD.id_lot = NEW.id_lot
        THEN
            RAISE EXCEPTION 'price не може бути нижче ніж
поточна !';
    END IF;
    IF NEW.price IS NULL THEN
        RAISE EXCEPTION ' price не може бути 0 ! ';
    END IF;
    RETURN NEW;
END
```



```
$update_bid_func$ LANGUAGE plpgsql;
```

Реалізували тригерну функцію для редагування таблиці ставок, код якої наведено у лістингу 3.4.

Дана функція оброблює інформацію про таблицю ставок та змінює статус інших ставок на «Відхилено», оскільки відредагована ставка може бути лише вищою за поточні. Також, вона змінює поточну ціну лоту на поточну.

Лістинг 3.4 — Запит на створення тригерної функції на зміну значень після модифікації таблиці ставок.

```
CREATE FUNCTION update_bid_af_func() RETURNS trigger AS
$update_bid_af_func $
BEGIN
    IF EXISTS( SELECT *
                FROM bid b
                WHERE b.price<NEW.price AND
b.id_lot=NEW.id_lot )
        THEN UPDATE bid SET bid_type='Refused'
WHERE bid.price<NEW.price AND bid.id_lot=NEW.id_lot;
        END IF;

    UPDATE lot SET current_price=NEW.price WHERE
NEW.id_lot=lot.id_lot;

    NEW.bid_type:='Accepted';
    RETURN NEW;
END;

$update_bid_af_func$ LANGUAGE plpgsql;
```

Також, створили тригерну функцію для видалення з таблиці ставок. Дана функція використовується для того, щоб уникнути ситуацій, коли актуальна ставка була видалена, а всі інші (що є цілком логічно) відхилені. В такому випадку функція обирає максимальну ставку з існуючих на даний лот на присвоює його поточній ціні значення максимальної ставки. Код тригерної функції наведено у лістингу 3.5.

Лістинг 3.5 — Запит на створення тригерної функції на видалення запису із таблиці ставок

```
CREATE FUNCTION del_func() RETURNS trigger AS
$del_func$

BEGIN

UPDATE bid SET bid_type='Accepted' WHERE price=(SELECT
MAX(price) FROM bid WHERE bid.id_lot=OLD.id_lot);

RETURN OLD;

$del_func$ LANGUAGE plpgsql;
```

Для таблиці лот також створили тригер на видалення, у якому реалізується переведення товарів лоту в статус проданий. Для цього була створена таблиця `sold_item`, опис якої наведено у таблиці 2.5. Дана функція перевіряє який статус був у видаленого лоту та у разі якщо він виявився зачиненим, то відбувається перенесення за допомогою допоміжних збережених процедур, код яких наведено у лістингах 3.6 — 3.7.

Лістинг 3.6 — Код для збереженої процедури, для отримання параметрів видаленого лоту

```
SELECT category_name, lot.current_price, now()::timestamp
without time zone, email_buyer, email_seller, lot.start_price ,
name_of_item
FROM lot
INNER JOIN item ON item.id_lot=idl
INNER JOIN "category" ON
"category".id_category=item.id_of_category
INNER JOIN seller ON item.id_seller=seller.id_seller
INNER JOIN bid ON idl=bid.id_lot AND bid.bid_type='Accepted'
INNER JOIN buyer ON buyer.id_buyer=bid.id_buyer
WHERE lot.id_lot=idl;
```

Лістинг 3.7 — Код для збереженої процедури, для створення проданого товару у спеціальній таблиці

```
INSERT INTO "sold_item" ("name_category", "final_price",
"data_of_buying", "email_seller", "email_buyer", "start_price",
"name_of_item")
VALUES ($1,$2,$3,$4,$5,$6,$7) RETURNING $7 ;
```

Для редагування лоту була реалізована тригерна функція, яка перевіряє логічні деталі: щоб початкова ціна не виявилася більшою за поточну, а також, щоб початок аукціону не виявився пізнішим за його кінець. Код даної функції наведено у лістингу 3.8.

Лістинг 3.8 — Код для тригерної функції, для перевірки коректності введених даних

```
BEGIN

    IF NEW.start_of_auction>NEW.end_of_auction

        THEN RAISE EXCEPTION 'Дата початку не може бути більша
за дату кінця !';

    END IF;

    IF NEW.start_price>NEW.current_price

        THEN RAISE EXCEPTION 'Початкова ціна не може бути нижче
поточної!';

    END IF;

    RETURN NEW;

END;
```

3.1.4 Перевірка ефективності доступу до даних

Оскільки в майбутньому наша система може зіштовхнутися з великим обсягом даних, то перевірили ефективності доступу даних. Заповнили БД великою кількістю даних:

- Bid — 2300000 записів;
- Buyers — 150050 записів;
- Lot — 200000 записів;
- Seller — 4 записів;
- Item — 1 запис;

- Category — 2 записи;
- SoldItem — 6 записів.

Після цього провели тестування та виявили, що використання , що використання Seq Scan є не дуже оптимізованим у випадку таблиці lot, тому створимо для неї індекс. Також, провели оптимізацію, результати якої надані у виді таблиць та графіків залежності для кожного кроку оптимізації.

Результати розрахунку середнього значення заміру часу виконання запиту після кожного кроку оптимізації представлені в таблиці 3.1 і на рисунках 3.9 — 3.10.

Розрахунок скорочення часу було виконано згідно формули за методичними рекомендаціями[6]:

$$f = (t_{\text{після}} - t_{\text{до}}) / t_{\text{до}} * 100 \quad (1.1)$$

де: f - приріст часу у відсотках;

$t_{\text{до}}$ - середній час виконання операції вставки даних до виконання кроку;

$t_{\text{після}}$ — середній час виконання після виконання кроку.

Таблиця 3.1 — Результати покрокового запуску запиту

Результати покрокової оптимізації				
		Ср.час виконання запиту до оптимізації	Ср.час виконання запиту після кроку	Скорочення часу виконання, %
К р о к	1	7932	7430	6,336933227
	2	7932	6863	13,48069084
	3	7932	4292	45,89233937
	4	7932	2733	65,54607724
	5	7932	2575	67,53792495
	6	7932	1540	80,58999033

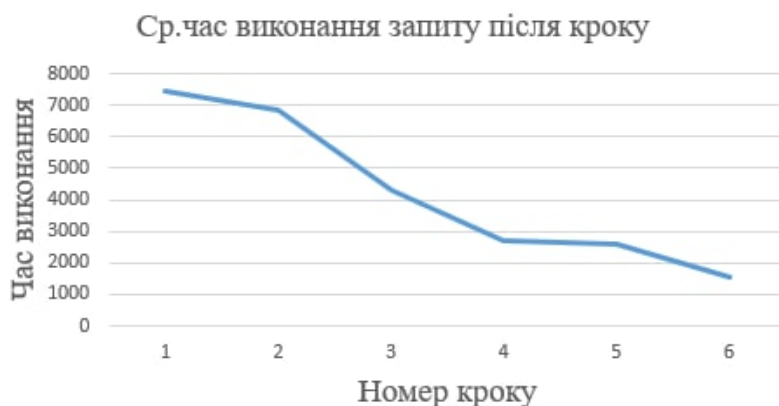


Рисунок 3.9 —Результати часу виконання запиту після кожного кроку

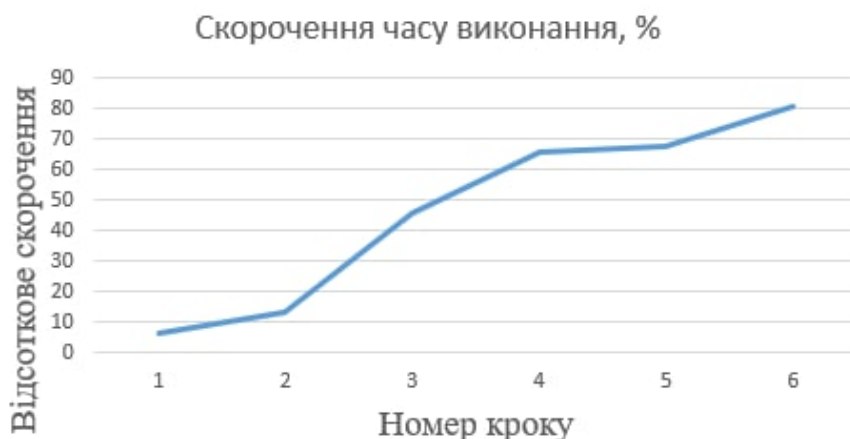


Рисунок 3.10 — Результати скорочення часу виконання запиту після кожного кроку

3.1.5 Підвищення надійності доступу до даних

Для реалізації механізму реплікації створили у БД користувача replication, який матиме можливість виконувати дії на обох серверах віддалено. Етап створення наведено на рисунку 3.11.

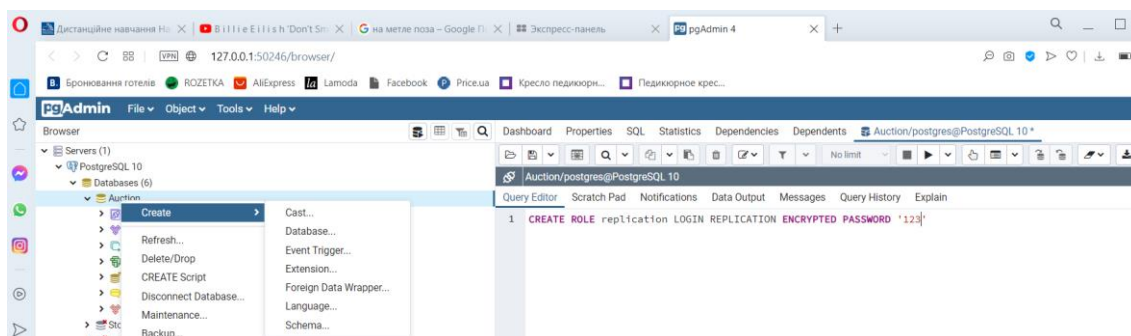


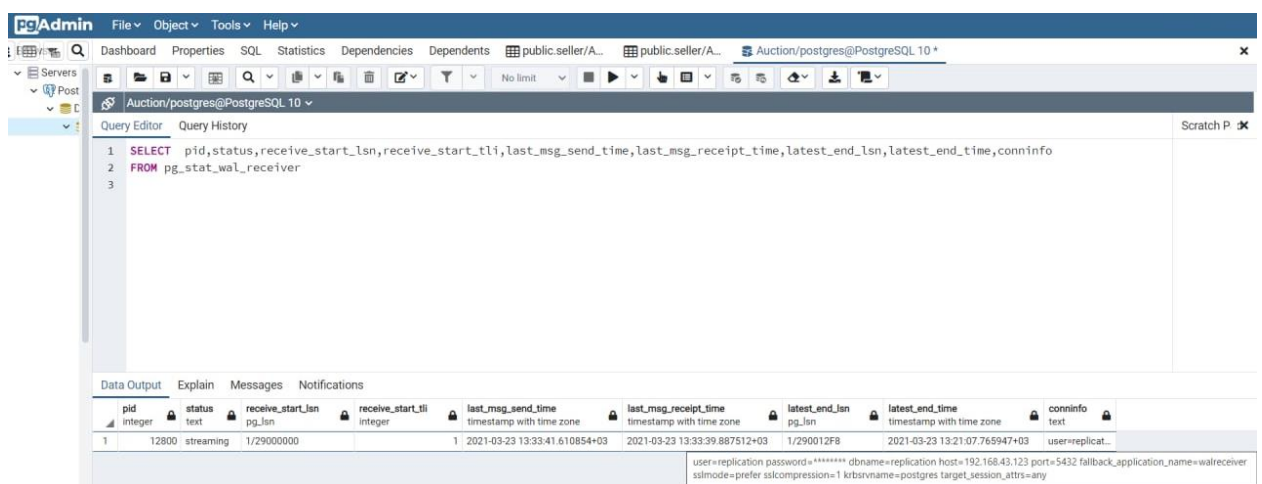
Рисунок 3.11 — Створення ролі для реплікації

Після чого створили слот, який буде використовуватися для реплікації з назвою `replica2` та провели тестування. На рисунках 3.12—3.13 наведена перевірка створеного слоту на майстрі та на слейві відповідно.

The screenshot shows the pgAdmin 4 web interface in a browser. The address bar displays '127.0.0.1:53612/browser/'. The top navigation bar includes 'pgAdmin', 'File', 'Object', 'Tools', and 'Help'. Below this is a breadcrumb trail: 'Dashboard > Properties > SQL > Statistics > Dependencies > Dependents > public.seller/Au... > Auction/postgres@PostgreSQL 10 *'. The main content area shows the 'Query Editor' tab for 'Auction/postgres@PostgreSQL 10'. The table 'pg_replication_slots' is displayed with the following data:

pid	usesysid	username	application_name	client_addr	client_port	backend_start	backend_xmin	?column?	sent_lsn	write_lsn	flush_lsn	replay_lsn	sync_priority
1	12160	66156	replication	walreceiver	192.168.43.15	58773	2021-03-23 13:10:53.972065+03	348964	state	1/290012F8	1/290012F8	1/290012F8	1/290012F8

Рисунок 3.12— Інформація про майстра



pid	status	receive_start_lsn	receive_start_tli	last_msg_send_time	last_msg_receipt_time	latest_end_lsn	latest_end_time	conninfo
1	streaming	1/29000000		1 2021-03-23 13:33:41.610854+03	2021-03-23 13:33:39.887512+03	1/290012F8	2021-03-23 13:21:07.765947+03	user=replicat...


```

1 SELECT pid,status,receive_start_lsn,receive_start_tli,last_msg_send_time,last_msg_receipt_time,latest_end_lsn,latest_end_time,conninfo
2 FROM pg_stat_wal_receiver
3

```

Рисунок 3.13— Інформація слейва

3.2 Розробка модулів системи

Система має 4 модулі серед яких:

- 1) Модуль шару доступу до даних;
- 2) Модуль шару бізнес-логіки;
- 3) Модуль сервісів;
- 4) Модуль відображення;

3.2.1 Розробка модулів шару бізнес логіки і бізнес правил

На рівні клієнта відбувається перевірка даних, які він отримав від шару сервісів та клієнта, а також обробка повідомлень від шару доступу до даних, на лістингу 3.9 наведено приклад перевірки даних отриманих від сервісів, а на лістингу 3.10 обробка результатів, отриманих від шару доступу до даних та передача їх до шару сервісів.

Лістинг 3.9 — Код для перевірки даних отриманих від сервісів

```

        try{
            if(buyer.getBuyerName().length() ==0
||buyer.getAddressBuyer().length()
==0||buyer.getPasswordBuyer().length()
==0||buyer.getPhoneNumberBuyer().length() ==0 ){
                req.setAttribute("error", "Error! You have empty
fields, please fulfill all of them and try again!");

req.getRequestDispatcher("addBuyer.jsp").forward(req, resp);
            }else if(!(buyer.getPhoneNumberBuyer().matches("[0-
9]+") && buyer.getPhoneNumberBuyer().length() >2) ){
                req.setAttribute("error", "Error! Phone number
should contains only numbers!");

req.getRequestDispatcher("addBuyer.jsp").forward(req, resp);
            }else {
                dao.saveUser(buyer);
            }
        }catch (Exception e){

            if(e.getMessage().contains("statement")){
                StringWriter errors = new StringWriter();
                e.printStackTrace(new PrintWriter(errors));

                String res =
Between(errors.toString(),"Подробности: Key ","exists.");
                req.setAttribute("error",
res.concat("exists."));
            }

req.getRequestDispatcher("addBuyer.jsp").forward(req, resp);
        }

    }
} catch (Exception e) {
    e.printStackTrace();}

```

Лістинг 3.10 — Код для обробки результатів, отриманих від шару доступу до даних та передача їх до шару сервісів

```

        buyersservice buy_dao=new buyersservice();
        List<buyer> buyerbs=buy_dao.findAllUsers();
double
av_b=(double)b_dao.findAllUsers().size()/(double)buyerbs.size();
        req.setAttribute("avg_buyer",four.format(av_b));

```

Ще однією функцією шару бізнес логіки є розмежування доступу до даних, система має 4 види користувачів, код проходження процедури автентифікації, ідентифікації наведено на лістингу 3.11.

Лістинг 3.11 — Фрагмент проходження процедур автентифікації та ідентифікації

```
if (req.getParameter("operation").equals("log")) {
    String email = req.getParameter("email");
    String password = req.getParameter("password");
    if (email.equals("postgres") &&
password.equals("qwel23rty456")) {
        HttpSession session1 = req.getSession(true);
        session1.setAttribute("admin", "admin");
        session1.setAttribute("id", "");
        String message = "Welcome admin!";
        req.setAttribute("message", message);
getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
forward(req, resp); }
```

В залежності від того, які права мають користувачі система вирішує яке повідомлення має отримати користувач, приклад перевірки наведено на лістингу 3.12

Лістинг 3.12— Фрагмент проходження авторизації

```
if (action != null && action.equals("delete")) {
    String param2 =
(String)req.getSession(false).getAttribute("admin");
    if(param2=="admin") {
        int id = Integer.parseInt(req.getParameter("id"));
        removeById(id);
        resp.sendRedirect("./bid");
    }else{
        String message = "You don`t have access to delete bids!";
        req.setAttribute("message", message);

getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
forward(req, resp);}
```

3.2.2 Розробка шару доступу до даних

Шар доступу до даних представляє собою набір класів, що реалізують запити до БД, написані на мові HQL. Розглянемо основні з них, які створені відносно кожної сутності. Код методів наведені у лістингах 3.13—3.17.

Лістинг 3.13 —Метод для отримання всіх покупців

```
public List<Object> findAll() {
    Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
    List users = (List<Object>)session.createQuery("from
buyerb").list(); }
```



```

    session.close();
    return users;
}

```

Лістинг 3.14 — Метод пошуку покупця за кодом

```

public buyerb findById(int id) {
    Session session =
    HibernateSessionFactoryUtil.getSessionFactory().openSession();
    buyerb b= session.get(buyerb.class, id);
    session.close();
    return b;}

```

Лістинг 3.15 — Метод видалення користувача

```

public void delete(Object obj) {
    Session session =
    HibernateSessionFactoryUtil.getSessionFactory().openSession();
    Transaction tx1 = session.beginTransaction();
    session.delete((buyerb) obj);
    tx1.commit();
    session.close();}

```

Лістинг 3.16 — Метод для редагування покупця

```

public void update(Object obj) {
    Session session =
    HibernateSessionFactoryUtil.getSessionFactory().openSession();
    Transaction tx1 = session.beginTransaction();
    session.update((buyerb) obj);
    tx1.commit();
    session.close();}

```

Лістинг 3.17 — Метод для додавання покупця

```

public void save(Object user) {
    Session session =
    HibernateSessionFactoryUtil.getSessionFactory().openSession();
    Transaction tx1 = session.beginTransaction();
    session.save((buyerb) user);
    tx1.commit();session.close();}

```

3.2.3 Розробка шару сервісів

Шар сервісів у системі представляють з себе набір додаткових функцій, які вона використовує. За натиском користувача на вікно статистики відбувається виклик сервісу Google Visualization API. Фрагмент коду для виклику сервісу за інтерфейсу наведено на лістингу 3.18.

Лістинг 3.18 — Фрагмент виклику сервісу з користувача

```

function drawChart() {
    var data = google.visualization.arrayToDataTable([
        ['Task', 'Page visit per Day'],
        ['Closed lots', ${closed}],
        ['Refused lots', ${refused}],
        ['Lots, that waiting for new bid', ${waiting}],
        ['Lots, for confirming', ${confirming}],
    ]);

    var options = {
        'width': 550,
        'height': 600,
        colors: ['#007BFF', '#00AEFF', '#00CCFF',
'#00EEFF'],
        pieHole: 0.4
    };
    var chart = new
google.visualization.PieChart(document.getElementById('piechart'
));
    chart.draw(data, options); }

```

Ще одним прикладом сервісу є надсилання листа на поштові скриньки продавця та остаточного покупця у разі успішного продажу товару, фрагмент коду, що демонструє виклик сервісу наведено у лістингу 3.19

Лістинг 3.19 — Фрагмент використання сервісу пошти

```

SendMail mailsender=new SendMail();
mailsender.sendSSLMessage(sendToseller.toArray(new String[0]),
"Results of Auction", "Congratulation! We have sold lot #" +
l_fordel.getIdLot()+" by price "+l_fordel.getCurrentPrice());
mailsender.sendSSLMessage(sendTobuyers.toArray( new String[0] ),
"Results of Auction", "Congratulation! We have bought lot #" +
l_fordel.getIdLot() +" by price "+ l_fordel.getCurrentPrice());

```

3.2.4 Розробка шару відображення

Шар відображення представлений у вигляді набору JSP-сторінок, для кожної сутності була створена сторінка додавання, редагування, а також перегляду усіх записів з можливістю видалення даних. Приклад таких сторінок наведено на рисунках 3.14—3.16.

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller

Search item by seller Search Login Logout

Price

Price

Status

Status of bid

Id buyer

dashaigor44@gmail.com ▾

Id lot

8193414 ▾

add

Рисунок 3.14 — Приклад форми додавання ставки

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller

Search item by seller Search Login Logout

Id

8193422

Current Price

99.0

Start Price

40.0

Status of lot

Waiting

Waiting for new bid ▾

Start of Auction

2021-08-01

End of Auction

2021-09-11

update

Рисунок 3.15 — Приклад форми редагування товару

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller

Search item by seller Search Login Logout

Our sold items

New sold item

Id	Final Price	Start Price	Name of category	E-mail of buyer	E-mail of seller	Name of item	Date of buying	Delete	Update
64214	99.0	50.0	Clothes	dashaigor44@gmail.com	mikaelladark1@gmail.com	PC-184	2021-05-11 13:29:28.317132	Remove	Update
64215	150.0	80.0	Computers	prischepa_darina@ukr.net	mikaelladark1@gmail.com	PC-190	2021-05-16 10:28:31.234943	Remove	Update

Рисунок 3.16 — Приклад форми списку проданих товарів

Для того, щоб надати користувачу пройти процедуру автентифікації було додану форму, яка наведена на рисунку 3.17. Для виходу з облікового запису достатньо натиснути кнопку «Logout», що викличе форму, що наведена на рисунку 3.18. Створена сторінка статистики наведена на рисунку 3.19.

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller

Search item by seller Search Login Logout

Enter your Data

E-mail

E-mail

Password

Password

log

[Sign in as buyer](#) [Sign in as seller](#)

Рисунок 3.17 — Приклад форми логіну

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller

Search item by seller Search Login Logout

You have logged out!

[To the items](#) [Login](#)

Рисунок 3.18 — Приклад форми виходу із системи

A little bit about our users

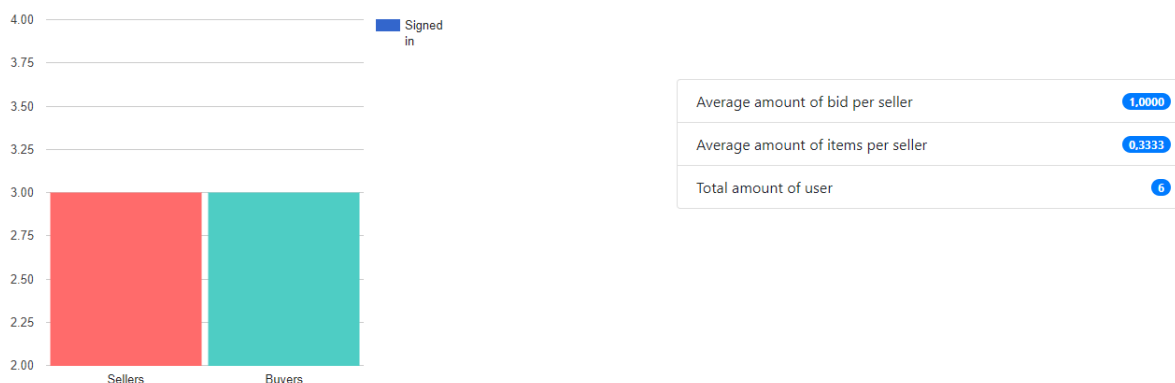


Рисунок 3.19 — Фрагмент форми статистики

3.3 Тестування створеної системи

Попередньо додали до бази даних, за допомогою pgAdmin, дані для тестування системи, результати тестування програми за різних випадків наведено на рисунках 3.20— 3.35.

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller

Search item by seller Search Login Logout

You don't have access to look at buyers!

[To the items](#) [Login](#)

Рисунок 3.20 — Виведення попередження до незареєстрованого користувача про обмеження у доступі до перегляду інформації

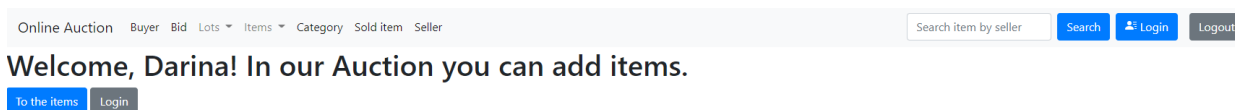


Рисунок 3.21 — Приклад авторизації користувача у вигляді продавця

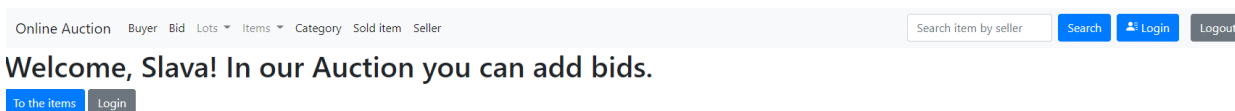


Рисунок 3.22 — Приклад авторизації користувача у вигляді покупця

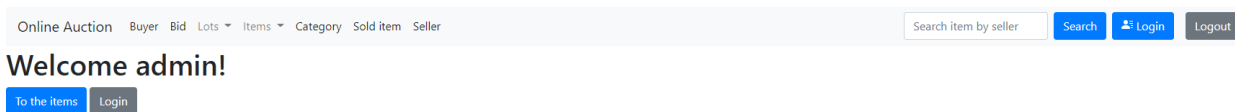


Рисунок 3.23 — Приклад авторизації користувача у вигляді адміністратора

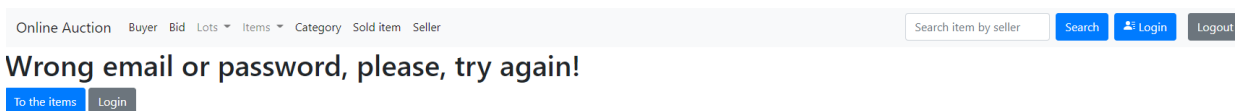


Рисунок 3.24 — Сповідження про введення некоректних даних користувача



Рисунок 3.25 — Приклад виходу із системи

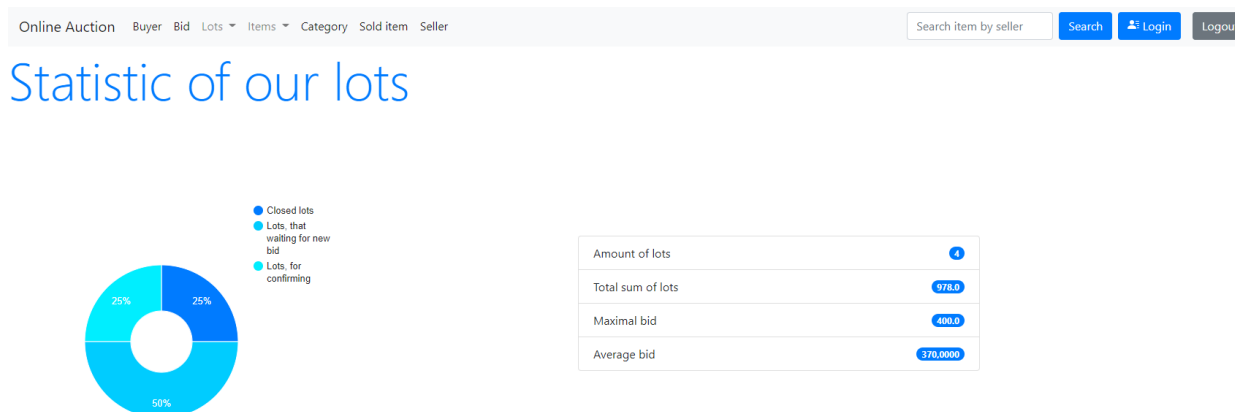


Рисунок 3.26 — Виведення статистики про лоти у системі

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller [Search](#) [Login](#) [Logout](#)

Our items

[New item](#)

Id	Name	Description	Seller	Category	Lot	Delete	Update
7	UserSP	Nice	1	14	8193411	Remove	Update

Рисунок 3.27 — Пошук товарів за категорії

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller [Search](#) [Login](#) [Logout](#)

Our items

[New item](#)

Id	Name	Description	Seller	Category	Lot	Delete	Update
7	UserSP	Nice	1	14	8193411	Remove	Update

Рисунок 3.28 — Пошук товарів за поштою продавця

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller [Search](#) [Login](#) [Logout](#)

Our lots

[New lot](#)

Id	Current Price	Start Price	Status of lot	Start of Auction	End of Auction	Delete	Update
8193419	99.0	70.0	Waiting for new bid	2021-08-01 00:00:00.0	2021-09-11 00:00:00.0	Remove	Update
8193422	99.0	40.0	Waiting for new bid	2021-08-01 00:00:00.0	2021-09-11 00:00:00.0	Remove	Update

Рисунок 3.29 — Пошук за статусом лоту

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller [Search](#) [Login](#) [Logout](#)

Our buyers

[New buyer](#)

Id	Name	Address	Password	Email	Phone	Delete	Update
10	Slava	Chernihiv	slava123slava	dashaigor44@gmail.com	0667149219	Remove	Update
9	Daryna	Chernihiv	qwe123rty	mikaelladark1@gmail.com	0937968974	Remove	Update
44	departure	Chernigov	slava123slava	dashaigor44@gmail.com	85654	Remove	Update

Рисунок 3.30 — Результат виведення списку покупців

Online Auction Buyer Bid Lots ▾ Items ▾ Category Sold item Seller [Search](#) [Login](#) [Logout](#)

Our buyers

[New buyer](#)

Id	Name	Address	Password	Email	Phone	Delete	Update
10	Slava	Chernihiv	slava123slava	dashaigor44@gmail.com	0667149219	Remove	Update
9	Daryna	Chernihiv	updatedpassword	mikaelladark1@gmail.com	0937968974	Remove	Update
46	Maksim	Chernigov	12345test12345	testuser@gmail.com	0935983465	Remove	Update

Рисунок 3.31 — Результат додавання покупця

Online Auction Buyer Bid Lots Items Category Sold item Seller							Search item by seller	Search	Login	Logout
Our buyers										
New buyer										
Id	Name	Address	Password	Email	Phone	Delete	Update			
10	Slava	Chernihiv	slava123slava	dashaigor44@gmail.com	0667149219	Remove	Update			
9	Daryna	Chernihiv	updatedpassword	mikaelladark1@gmail.com	0937968974	Remove	Update			

Рисунок 3.32 — Результат редагування покупця

Online Auction Buyer Bid Lots Items Category Sold item Seller							Search item by seller	Search	Login	Logout
Our buyers										
New buyer										
Id	Name	Address	Password	Email	Phone	Delete	Update			
10	Slava	Chernihiv	slava123slava	dashaigor44@gmail.com	0667149219	Remove	Update			
9	Daryna	Chernihiv	qwe123rty	mikaelladark1@gmail.com	0937968974	Remove	Update			

Рисунок 3.33 — Результат видалення покупця

Online Auction

Buyer

Bid

Lots

Items

Category

Sold item

Seller

Search item by seller

Search

Login

Logout

Our sold items

New sold item

Id	Final Price	Start Price	Name of category	E-mail of buyer	E-mail of seller	Name of item	Date of buying	Delete	Update
64214	99.0	50.0	Clothes	dashaigor44@gmail.com	mikaelladark1@gmail.com	PC-184	2021-05-11 13:29:28.317132	Remove	Update
64215	150.0	80.0	Computers	prischepa_darina@ukr.net	mikaelladark1@gmail.com	PC-190	2021-05-16 10:28:31.234943	Remove	Update

Рисунок 3.34 — У результаті видалення лоту зі статусом «Closed» додався новий запис у таблиці sold_item



Рисунок 3.35 — У разі успішного продажу товару, покупцю та продавцю приходить наступне повідомлення на пошту

3.4 Висновки до розділу 3

За спроектованими рівнями у розділі 2 реалізували систему Онлайн-Аукціону, розробили основні механіки реалізації бізнес-логіки системи (тригери, збережені процедури), виявили особливості роботи системи при високому навантаженні, створили індекс для таблиці лотів, розробили шар сервісів, зокрема передачу повідомлень, а також реалізували шар відображення. Для покращення надійності був розроблений механізм реплікації. Протестували розроблену систему з метою виявлення специфіки її використання.

ВИСНОВКИ

За результатами виконання курсової роботи сформовано наступні висновки:

1) Під час розробки курсового проекту була створена повноцінна багаторівнева система, яка демонструє роботу онлайн-аукціону, з урахуванням особливостей зберігання даних.

2) Створена система складається з чотирьох рівнів: база даних, доступ до даних, бізнес-логіка, сервіси, інтерфейс. Кожен з них використовує унікальні для системи технології. Так, рівень баз даних був розроблений за допомогою PostgreSQL, доступ до даних на ORM-технології, сервіси за допомогою JavaxMail та Google Visualization API, бізнес-логіка була написана мовою JAVA у поєднанні з стандартизованим API Javax.Servlet. Для шару відображення було використано велику кількість технологій, основною з яких стала JSP.

3) Протестували створену програму і виявили, що вона працює цілком адекватно і відповідає поставленій задачі.

4) Завдяки тому, що система є багаторівневою є можливість розширювати її незалежно від інших рівнів, наприклад, розглядається удосконалення системи для зберігання зображень у якості додаткового атрибута таблиці item.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Що таке база даних.[Електронний ресурс].—Режим доступу: <http://apeps.kpi.ua/shco-take-basa-danykh>.
2. Эрик Редмонд, Джим. Р. Уилсон. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL = MongoDB in Action. — ДМК Пресс, 2013. — 384 с. — ISBN 978-5-94074-866-3.
3. Сравнение MySQL и PostgreSQL.[Електронний ресурс].— Режим доступу: <https://losst.ru/sravnenie-mysql-i-postgresql>.
4. Часть 5. Сервлеты, Java servlet API. Пишем простое веб-приложение [Електронний ресурс].— Режим доступу: <https://javarush.ru/groups/posts/2529-chastjh-5-servletih-pishem-prostoe-veb-prilozhenie>.
5. CSS. [Електронний ресурс].— Режим доступу: <https://ru.m.wikipedia.org/wiki/CSS>.
6. Оптимізація SQL-запитів. Методичні вказівки до виконання розрахунково-графічної роботи з дисципліни «Бази даних» для студентів напряму підготовки 121 – „Інженерія програмного забезпечення”. /Укл.: Білоус І.В. – ЧНТУ, 2020. – 15с. – Електронні данні – Режим доступу: <https://eln.stu.cn.ua/course/view.php?id=802> , обмежений. – Заголовок з екрану.

ДОДАТКИ

Додаток А
«Лістинги застосунку»

Лістинг файлу SecurityController.java

```
package com.mandarinka.lab5.controllers;

import com.mandarinka.lab5.models.buyerb;
import com.mandarinka.lab5.models.category;
import com.mandarinka.lab5.models.seller;
import com.mandarinka.lab5.services.buyersservice;
import com.mandarinka.lab5.services.categoryservice;
import com.mandarinka.lab5.services.sellerservice;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.List;

public class SecurityController extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req,
        HttpServletResponse resp)
        throws ServletException, IOException {

        categoryservice cat_dao=new categoryservice();
        List<category> cats=cat_dao.findAllUsers();
        req.setAttribute("cats", cats);

        String action = req.getParameter("action");
        if (action != null && action.equals("logout")) {
            String message = "You have logged out!";
            req.setAttribute("message", message);
            HttpSession session1 = req.getSession(true);
            session1.setAttribute("admin", null);

            getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
            forward(req, resp);
            } else {

            getServletContext().getRequestDispatcher("/SecurityLogin.jsp").f
            orward(req, resp);
            }

    }

    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse resp)
        throws ServletException, IOException {
```

```

        categoryservice cat_dao=new categoryservice();
        List<category> cats=cat_dao.findAllUsers();
        req.setAttribute("cats", cats);

        if (req.getParameter("operation").equals("log")) {
            String email = req.getParameter("email");
            String password = req.getParameter("password");
            if (email.equals("postgres") &&
password.equals("qwel23rty456")) {
                HttpSession session1 = req.getSession(true);
                session1.setAttribute("admin", "admin");
                session1.setAttribute("id", "");
                String message = "Welcome admin!";
                req.setAttribute("message", message);

getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
forward(req, resp);
            } else {
                buyersservice dao=new buyersservice();
                buyerb buyer=dao.validateUser(email,password);
                if(buyer!=null){
                    HttpSession session1 = req.getSession(true);
                    session1.setAttribute("admin", "buyer");

session1.setAttribute("id",buyer.getIdBuyer());

                    String message = "Welcome,
"+buyer.getBuyerName()+ "! \n In our Auction you can add bids."
;
                    req.setAttribute("message", message);

getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
forward(req, resp);
                }else{
                    sellerservice daosel=new sellerservice();
                    seller
sel=daosel.validateUser(email,password);

                    if(sel!=null){
                        HttpSession session1 =
req.getSession(true);
                        session1.setAttribute("admin", "seller");

session1.setAttribute("id",sel.getIdSeller());

                        String message = "Welcome,
"+sel.getNameSeller()+ "! \n In our Auction you can add items." ;
                        req.setAttribute("message", message);

getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
forward(req, resp);
                    } else{

```

```

        HttpSession session1 =
req.getSession(true);
        session1.setAttribute("admin", null);
        String message = "Wrong email or password,
please, try again!" ;
        req.setAttribute("message", message);

getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
forward(req, resp);

    }
}

}
}
}else{
    resp.sendRedirect("./security");
} }

```

Лістинг файлу ServletHomeBid.java

```

package com.mandarinka.lab5.controllers;

import com.mandarinka.lab5.models.bid;
import com.mandarinka.lab5.models.buyerb;
import com.mandarinka.lab5.models.category;
import com.mandarinka.lab5.models.lot;
import com.mandarinka.lab5.services.bidservice;
import com.mandarinka.lab5.services.buyersservice;
import com.mandarinka.lab5.services.categoryservice;
import com.mandarinka.lab5.services.lotservice;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.util.List;

public class ServletHomeBid extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {

        bidservice dao = new bidservice();
        categoryservice cat_dao=new categoryservice();
        List<category> cats=cat_dao.findAllUsers();

        req.setAttribute("cats", cats);
        String action = req.getParameter("action");
        PrintWriter out = resp.getWriter();
        out.println("<script type="+ "text/javascript"+ ">");

```

```

        out.println("alert(\"+\"Hello"+"");");
        out.println("</script>");
        if (action != null && action.equals("delete")) {
            String param2 =
(String)req.getSession(false).getAttribute("admin");
            if(param2=="admin") {
                int id =
Integer.parseInt(req.getParameter("id"));
                removeById(id);
                resp.sendRedirect("./bid");
            }else{
                String message = "You don`t have access to delete
bids!";
                req.setAttribute("message", message);

getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
forward(req, resp);
            }

        } else if (action != null && action.equals("update")) {
            String param2 =
(String)req.getSession(false).getAttribute("admin");
            System.out.println("Param: "+param2);
            if(param2=="admin") {
                int id =
Integer.parseInt(req.getParameter("id"));
                bid uBid = dao.findUser(id);
                if (uBid != null) {
                    buyersservice bdao = new buyersservice();
                    lotservice ldao=new lotservice();
                    req.setAttribute("uBid", uBid);
                    List<buyer> buyers=bdao.findAllUsers();
                    List<lot> lots=ldao.findAllUsers();
                    req.setAttribute("buyers", buyers);
                    req.setAttribute("lots", lots);

getServletContext().getRequestDispatcher("/updateBid.jsp").forwa
rd(req, resp);
                }
            }
            else{
                String message = "You don`t have access to update
bids!";
                req.setAttribute("message", message);

getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
forward(req, resp);
            }
        }else if (action != null && action.equals("add")) {
            String param2 =
(String)req.getSession(false).getAttribute("admin");

            if(param2=="admin" || param2=="buyer") {

```



```

        buyersservice bdao = new buyersservice();
        lotservice ldao=new lotservice();
        String param3 =
String.valueOf(req.getSession(false).getAttribute("id"));
        List<buyer> buyers=bdao.findAllUsers();
        List<lot> lots=ldao.findAllUsers();
        req.setAttribute("buyers", buyers);
        req.setAttribute("lots", lots);
        req.setAttribute("id", param3);

getServletContext().getRequestDispatcher("/addBid.jsp").forward(
req, resp);
        }else{
            String message = "You don`t have access to add
bids!";
            req.setAttribute("message", message);

getServletContext().getRequestDispatcher("/SecurityDenied.jsp").
forward(req, resp);
        }
        }else{
            List<bid> bids = dao.findAllUsers();
            req.setAttribute("bids", bids);

getServletContext().getRequestDispatcher("/ListBid.jsp").forward
(req, resp);
        }
    }

    @Override
    protected void doPost(HttpServletRequest req,
    HttpServletResponse resp)
        throws ServletException, IOException {
        if (req.getParameter("operation").equals("add")) {
            bid nBid = new bid();
            lotservice s_lot=new lotservice();
            buyersservice s_buyer=new buyersservice();
            try {

nBid.setPrice(Double.parseDouble(req.getParameter("price")));
                nBid.setStatusBid(req.getParameter("status"));

nBid.setLotByIdLot(s_lot.findUser(Integer.parseInt(req.getParame
ter("lot"))));

nBid.setBuyersByIdBuyer(s_buyer.findUser(Integer.parseInt(req.ge
tParameter("buyer"))));
                bidservice dao = new bidservice();

                dao.saveUser(nBid);

```

```

        } catch (Exception e) {
            if(e.getMessage().contains("empty")) {
                req.setAttribute("error", "Error! You have
empty fields, please fulfill all of them and try again!");

                }else if(e.getMessage().contains("statement")){
                    StringWriter errors = new StringWriter();
                    e.printStackTrace(new PrintWriter(errors));

                    String res =
Between(errors.toString(), "org.postgresql.util.PSQLException:", "
Где:");

                    System.out.println("Error123 "+res);
                    req.setAttribute("error", res);
                }else if(e.getMessage().contains("For input
string")){
                    req.setAttribute("error", "Error! You have
data with wrong format, please, make sure that your prices entered
in format 0.00 and your dates in format yyyy-mm-dd hh:mm:ss.ms!");
                }
                buyersservice bdao = new buyersservice();
                lotservice ldao=new lotservice();
                List<buyer> buyers=bdao.findAllUsers();
                List<lot> lots=ldao.findAllUsers();
                req.setAttribute("buyers", buyers);
                req.setAttribute("lots", lots);

req.getRequestDispatcher("addBid.jsp").forward(req, resp);
                e.printStackTrace();
            }
        } else if
(req.getParameter("operation").equals("update")) {
            bidservice dao = new bidservice();
            lotservice s_lot=new lotservice();
            buyersservice s_buyer=new buyersservice();
            bid uBid =
dao.findUser(Integer.parseInt(req.getParameter("id")));

            try {

uBid.setPrice(Double.parseDouble(req.getParameter("price")));
                uBid.setStatusBid(req.getParameter("status"));

uBid.setLotByIdLot(s_lot.findUser(Integer.parseInt(req.getParame
ter("lot"))));

uBid.setBuyersByIdBuyer(s_buyer.findUser(Integer.parseInt(req.ge
tParameter("buyer"))));

            } catch (Exception e) {

                if(e.getMessage().contains("empty")) {

```

```

        req.setAttribute("error", "Error! You have
empty fields, please fulfill all of them and try again!");

    }else    if(e.getMessage().contains("For    input
string")){
        req.setAttribute("error", "Error! You have
data with wrong format, please, make sure that your prices entered
in format 0.00 and your dates in format yyyy-mm-dd hh:mm:ss.ms!");
    }
    buyersservice bdao = new buyersservice();
    lotservice ldao=new lotservice();
    List<buyer> buyers=bdao.findAllUsers();
    List<lot> lots=ldao.findAllUsers();
    req.setAttribute("buyers", buyers);
    req.setAttribute("lots", lots);
    req.setAttribute("uBid", uBid);

req.getRequestDispatcher("updateBid.jsp").forward(req, resp);
    }
    try{

        dao.updateUser(uBid);
    }catch (Exception e1){
        if(e1.getMessage().contains("statement")){

            StringWriter errors = new StringWriter();
            e1.printStackTrace(new PrintWriter(errors));
            e1.printStackTrace();
            String                res                =
Between(errors.toString(),"org.postgresql.util.PSQLException:", "
Где:");

            System.out.println("Error123 "+res);
            req.setAttribute("error", res);
        }
        buyersservice bdao = new buyersservice();
        lotservice ldao=new lotservice();
        List<buyer> buyers=bdao.findAllUsers();
        List<lot> lots=ldao.findAllUsers();
        req.setAttribute("buyers", buyers);
        req.setAttribute("lots", lots);

        req.setAttribute("uBid", uBid);

req.getRequestDispatcher("updateBid.jsp").forward(req, resp);
    }
}

resp.sendRedirect("./bid");
}
public String Between(String STR , String FirstString, String
LastString)
{
    String FinalString;

```

```

        int Pos1 = STR.indexOf(FirstString) +
FirstString.length();
        int Pos2 = STR.indexOf(LastString);
        System.out.println("Pos1 " + Pos1);
        System.out.println("Pos2 "+Pos2);
        FinalString = STR.substring(Pos1, Pos2);
        return FinalString;
    }
    private void removeById(int id) {
        bidservice dao = new bidservice();
        dao.deleteUser(dao.findUser(id));
    }
}

```

Лістинг файлу ServletOfStatistic.java

```

package com.mandarinka.lab5.controllers;

import com.mandarinka.lab5.models.*;
import com.mandarinka.lab5.services.*;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class ServletOfStatistic extends HttpServlet {

    @Override

    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {

        lotservice dao=new lotservice();

        String closed = "Closed";

        String refused="Refused";

        String waiting="Waiting for new bid";

        String confirming="Waiting for confirming";

        List<lot> clos_lots = dao.findAlllotsbystatus(closed);
    }
}

```

```

        List<lot>                                refused_lots                =
dao.findAlllotsbystatus(refused);

        List<lot>                                waiting_lots               =
dao.findAlllotsbystatus(waiting);

        List<lot>                                conf_lots                  =
dao.findAlllotsbystatus(confirming);


req.setAttribute("closed", clos_lots.size());
req.setAttribute("waiting", waiting_lots.size());
req.setAttribute("refused", refused_lots.size());
req.setAttribute("confirming", conf_lots.size());


List<Integer> data= new ArrayList<>();
categoryservice dao_cat= new categoryservice();
List<category> cats=dao_cat.findAllUsers();
itemservice dao_it=new itemservice();
for (int i=0; i<cats.size();i++){
    category cat=cats.get(i);

data.add(dao_it.findAllUsersbyCat(cat.getIdOfCategory()).size())
;

    }


req.setAttribute("item_n",data);
req.setAttribute("cats",cats );


req.setAttribute("amount_of_item",dao_it.findAllUsers().size());

req.setAttribute("amount_of_lots",dao.findAllUsers().size());
    req.setAttribute("sum_of_lots",dao.sumAll());

```

```

        solditemservice s_dao=new solditemservice();

req.setAttribute("amount_of_sold_item",s_dao.findAllUsers().size() );

req.setAttribute("total_sum_of_solditems",s_dao.sumAll());
        req.setAttribute("last_sold_item",s_dao.lastSold());


        DecimalFormat four = new DecimalFormat("#0.0000");
        bidservice b_dao=new bidservice();
        req.setAttribute("max_bid",b_dao.maxBid());
        req.setAttribute("avg_bid",four.format(b_dao.avgAll()));


        buyersservice buy_dao=new buyersservice();
        List<buyer> buyerbs=buy_dao.findAllUsers();

double
av_b=(double)b_dao.findAllUsers().size()/(double)buyerbs.size();
        req.setAttribute("avg_buyer",four.format(av_b));
        sellerservice sel_dao=new sellerservice();
        List<seller> sellers=sel_dao.findAllUsers();


        double
av_s=(double)dao_it.findAllUsers().size()/(double)sellers.size()
;

req.setAttribute("avg_seller",four.format(av_s));
        req.setAttribute("total_buyers",buyerbs.size());
        req.setAttribute("total_sellers",sellers.size());


getServletContext().getRequestDispatcher("/Statistic.jsp").forward(req, resp);

    }

@Override

```

```

        protected void doPost(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {

        }

    }
}

```

Лістинг файлу DaoInterface.java

```

package com.mandarinka.lab5.dao;
import java.util.List;
public interface DaoInterface {
    void save(Object obj);
    Object findById(int id);
    void update(Object obj);
    void delete(Object obj);
    List<Object> findAll();
}

```

Лістинг файлу sellerDaoImpl.java

```

package com.mandarinka.lab5.dao;
import com.mandarinka.lab5.models.seller;
import com.mandarinka.lab5.utils.HibernateSessionFactoryUtil;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.criterion.Restrictions;
import java.util.List;

public class sellerDaoImpl implements DaoInterface {
    public void save(Object user) {
        Session session =
        HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.save((seller)user);
        tx1.commit();
        session.close();
    }

    public seller findById(int id) {
        Session session =
        HibernateSessionFactoryUtil.getSessionFactory().openSession();
        seller b= session.get(seller.class, id);
        session.close();
        return b;
    }

    public void update(Object obj) {

```

```

        Session                                session                                =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.update((seller) obj);
        tx1.commit();
        session.close();
    }

    public void delete(Object obj) {
        Session                                session                                =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.delete((seller) obj);
        tx1.commit();
        session.close();
    }

    public List<Object> findAll() {

        Session                                session                                =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        List sellers = (List<Object>)session.createQuery("from
seller").list();
        session.close();
        return sellers;
    }

    public seller validate(String email, String password) {
        Session                                session                                =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Criteria                                crit=
session.createCriteria(seller.class).add(Restrictions.eq("eMailS
eller", email)).add(Restrictions.eq("passwordSeller", password));
        seller sel=(seller) crit.uniqueResult();
        session.close();
        return sel;
    }

    public seller findemail(String email) {
        Session                                session                                =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Criteria                                crit=
session.createCriteria(seller.class).add(Restrictions.eq("eMailS
eller", email));
        seller sel=(seller) crit.uniqueResult();
        session.close();
        return sel;
    }
}

```


Лістинг файлу solditem.java

```
package com.mandarinka.lab5.models;
import javax.persistence.*;
import java.sql.Timestamp;
@Entity
@Table(name = "sold_item", schema = "public", catalog =
"Auction")
public class solditem {
    private int idSoldItem;
    private String nameCategory;
    private Double finalPrice;
    private Timestamp dataOfBuying;
    private String eMailSeller;
    private String eMailBuyer;
    private Double startPrice;
    private String nameOfItem;
    private item itemByIdItem;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_sold_item")
    public int getIdSoldItem() {
        return idSoldItem;
    }

    public void setIdSoldItem(int idSoldItem) {
        this.idSoldItem = idSoldItem;
    }

    @Basic
    @Column(name = "name_category")
    public String getNameCategory() {
        return nameCategory;
    }

    public void setNameCategory(String nameCategory) {
        this.nameCategory = nameCategory;
    }

    @Basic
    @Column(name = "final_price")
    public Double getFinalPrice() {
        return finalPrice;
    }

    public void setFinalPrice(Double finalPrice) {
        this.finalPrice = finalPrice;
    }

    @Basic
    @Column(name = "data_of_buying")
    public Timestamp getDataOfBuying() {
```

```

        return dataOfBuying;
    }

    public void setDataOfBuying(Timestamp dataOfBuying) {
        this.dataOfBuying = dataOfBuying;
    }

    @Basic
    @Column(name = "email_seller")
    public String geteMailSeller() {
        return eMailSeller;
    }

    public void seteMailSeller(String eMailSeller) {
        this.eMailSeller = eMailSeller;
    }

    @Basic
    @Column(name = "email_buyer")
    public String geteMailBuyer() {
        return eMailBuyer;
    }

    public void seteMailBuyer(String eMailBuyer) {
        this.eMailBuyer = eMailBuyer;
    }

    @Basic
    @Column(name = "start_price")
    public Double getStartPrice() {
        return startPrice;
    }

    public void setStartPrice(Double startPrice) {
        this.startPrice = startPrice;
    }

    @Basic
    @Column(name = "name_of_item")
    public String getNameOfItem() {
        return nameOfItem;
    }

    public void setNameOfItem(String nameOfItem) {
        this.nameOfItem = nameOfItem;
    }
}

```

Лістинг файлу lotservice.java

```
package com.mandarinka.lab5.services;
```

```

import com.mandarinka.lab5.models.lot;
import com.mandarinka.lab5.dao.lotDaoImpl;

import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
import java.util.List;

public class lotservice {
    private lotDaoImpl usersDao = new lotDaoImpl();
    public lotservice() {
    }
    public lot findUser(int id) {
        return usersDao.findById(id);
    }

    public void saveUser(lot user) {
        usersDao.save(user);
    }

    public void deleteUser(lot user) {
        usersDao.delete(user);
    }

    public void updateUser(lot user) {
        usersDao.update(user);
    }

    public List<lot> findAllUsers() {
        return (List<lot>) (List<?>) usersDao.findAll();
    }

    public List<lot> findAlllotsbystatus(String status) {
        return (List<lot>) (List<?>)
usersDao.findByStatus(status);
    }

    public Double sumAll(){return usersDao.sumAll();}


    public DefaultTableModel tablemodelLot(){
        List<lot> tablelist= findAllUsers();
        DefaultTableModel model = new DefaultTableModel();
        model.setColumnIdentifiers(new String[]{"ID lot", "Id
bid", "Start Price", "Current Price", "Start Of Auction", "End Of
Auction", "Status of Lot"});
        for(int i=0;i<tablelist.size();i++){
            ArrayList<String> row = new ArrayList<>();

row.add(String.valueOf(tablelist.get(i).getIdLot()));

```

```

row.add(String.valueOf(tablelist.get(i).getBidsByIdLot()));
row.add(String.valueOf(tablelist.get(i).getStartPrice()));
row.add(String.valueOf(tablelist.get(i).getCurrentPrice()));
row.add(String.valueOf(tablelist.get(i).getStartOfAuction()));
row.add(String.valueOf(tablelist.get(i).getEndOfAuction()));
row.add(String.valueOf(tablelist.get(i).getStatusLot()));
        model.addRow(row.toArray());
    }
    return model;
}
}

```

Лістинг файлу MainServlet.java

```

package com.mandarinka.lab5;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;

public class MainServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req,
        HttpServletResponse resp)
        throws ServletException, IOException {
        String param1 = req.getParameter("admin");
        HttpSession session1 = req.getSession(true);
        session1.setAttribute("admin", param1);
        resp.sendRedirect("./statistic");
    }

    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse resp)
        throws ServletException, IOException {
        super.doPost(req, resp);
    }
}

```

```
}
```

Лістинг файлу hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="connection.url">jdbc:postgresql://localhost:5432/Auction</
property>
        <property
name="connection.driver_class">org.postgresql.Driver</property>
        <!-- <property name="connection.username"/> -->
        <property
name="hibernate.connection.username">postgres</property>
        <!-- <property name="connection.password"/> -->
        <property
name="hibernate.connection.password">9810</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.PostgreSQL10Diale
ct</property>
        <property name="show_sql">true</property>
        <!-- DB schema will be updated if needed -->
        <!-- <property
name="hibernate.hbm2ddl.auto">update</property> -->
    </session-factory>
</hibernate-configuration>
```

Лістинг файлу addBuyer.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
    <title>Buyer</title>
```

```

    <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
</head>
<body>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootst
rap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
" crossorigin="anonymous">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootst
rap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
" crossorigin="anonymous">
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN
" crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd
/popper.min.js" integrity="sha384-
b/U6ypiBEHpOf/4+1nzFpr53nxSS+GLCKfwbDfNtXtclqgenISfwAzpKaMnFNMj4
" crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta/js/bootstrap.min.js" integrity="sha384-
h0AbiXch4ZDo7tp9hKZ4TsHbi047NrKGL03SEJAg45jXxnGIIfYzk4Si90RDIqNm1
" crossorigin="anonymous"></script>

<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand " href="./statistic">Online
Auction</a>
    <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse "
id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="./buyer">Buyer</a>
            </li>

            <li class="nav-item active">
                <a class="nav-link" href="./bid">Bid</a>
            </li>

```

```

        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="/lot"
id="navbarDropdownMenuLink" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">
                Lots
            </a>
            <div class="dropdown-menu" aria-
labelledby="navbarDropdownMenuLink">
                <a class="dropdown-item"
href="/lot?action=specialstat&status="Waiting%20for%20new%20bid"
>Waiting for new bid</a>
                <a class="dropdown-item"
href="/lot?action=specialstat&status=Closed">Closed</a>
                <a class="dropdown-item"
href="/lot?action=specialstat&status=Refused">Refused</a>
                <a class="dropdown-item"
href="/lot?action=specialstat&status="Waiting%20for%20confirming
">Waiting for confirming</a>
                <div class="dropdown-divider"></div>
                <a class="dropdown-item"
href="/lot?action=allitems">All lots</a>
            </div>
        </li>

        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle"
id="navbarDropdown" role="button" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">
                Items
            </a>
            <div class="dropdown-menu" aria-
labelledby="navbarDropdown">

                <c:forEach var="cat" items="${cats}">
                    <a class="dropdown-item"
href="/item?action=speciatcat&id_cat=${cat.getIdOfCategory()}>${
cat.getCategoryName()}</a>
                </c:forEach>
                <div class="dropdown-divider"></div>
                <a class="dropdown-item"
href="/item?action=allitems">All items</a>
            </div>

        </li>
        <li class="nav-item active">
            <a class="nav-link"
href="/category">Category</a>
        </li>

```

```

        <li class="nav-item active">
            <a class="nav-link" href="./solditem">Sold
item</a>
        </li>

        <li class="nav-item active">
            <a class="nav-link" href="./seller">Seller</a>
        </li>
    </ul>
    <form class="form-inline my-2 my-lg-0" method="POST"
action="./item">

        <input class="form-control mr-sm-2" type="search"
name=searchsel placeholder="Search item by seller" aria-
label="Search">

        <button class="btn btn-primary my-2 my-sm-0"
name="operation" type="submit" value="sel_ser">Search</button>

    </form>

    <form class="form-inline my-2 my-lg-0" method="POST"
action="./security">
        <button class="btn btn-primary mr-2 ml-2"
name="operation" type="submit" value="form">
            <svg xmlns="http://www.w3.org/2000/svg" width="16"
height="16" fill="currentColor" class="bi bi-person-lines-fill"
viewBox="0 0 16 16">
                <path d="M6 8a3 3 0 1 0 0-6 3 3 0 0 0 0 6zm-
5 6s-1 0-1-1 1-4 6-4 6 3 6 4-1 1-1 1H1zM11 3.5a.5.5 0 0 1 .5-
.5h4a.5.5 0 0 1 0 1h-4a.5.5 0 0 1-.5-.5zm.5 2.5a.5.5 0 0 0 0
1h4a.5.5 0 0 0 0-1h-4zm2 3a.5.5 0 0 0 0 1h2a.5.5 0 0 0 0-1h-2zm0
3a.5.5 0 0 0 0 1h2a.5.5 0 0 0 0-1h-2z"/>
            </svg>
            Login
        </button>
    </form>
    <a class="btn btn-secondary mr-2 ml-2 "
href=./security?action=logout>Logout</a>
</div>

</nav>

<form method="post">

<div class="form-group" style=width:50%>
    <p class="text-center" style="color: #007bff">${error}</p>
    <label>Name</label>

```



```
<input type="text" class="form-control" name=name
placeholder=Name>

<label>E-mail</label>
<input type="text" class="form-control" name=email
placeholder=E-mail>

<label>Password</label>
<input type="text" class="form-control" name=password
placeholder=Password>

<label>Address</label>
<input type="text" class="form-control" name=address
placeholder=Address>

<label>Phone</label>
<input type="text" class="form-control" name=phone
placeholder=Phone>

<input class="btn btn-primary" name=operation type="submit"
value="add" /><br>
</div>

</form>
</body>
</html>
```