

ВСТУП

Сьогодні серед населення спостерігається тенденція скорочення вільного часу. Часом членам будь-яких верств населення невістачає часу на дійсно приємні чи навіть важливі речі. Саме цю проблему вирішує даний проект.

Продукт має реальну можливість утвердження на ринку.

Програмний продукт візьме на себе роль посередника між людьми, яким повинно виконати якесь завдання, та людьми, які таку можливість мають. Це можуть бути ремесленники, репетитори та просто люди у яких на це вистачає вільного часу.

Аналоги таких систем час від часу з'являлися на європейських та американських ринках. Іноді навіть на вітчизняних.

Розробка данного проекту забезпечить можливість користувачам заощаджувати на власному часі. В наслідок цього витратити більше часу на саморозвиток, розваги і так далі.

За допомогою об'єктно-орієнтованого підходу програмний продукт схожий на опис об'єктів їх характеристик, а також сукупностей класів, відношенню між ними, способів їх взаємодії та операцій над об'єктами. Об'єктно-орієнтоване програмування в даний час є абсолютним лідером області прикладного програмування, за допомогою цього підходу дуже легко і швидко створювати програми, котрі дуже просто підтримуються.

Метою курсової роботи є проектування, розробка та тестування програмного проекту з використанням рекомендацій та бібліотек Spring MVC[1], Android; самостійне вивчення принципів написання чистого коду, основ рефакторінгу та реінжинірингу коду, закріплення теоретичних знань та практичних навичок, отриманих в ході вивчення даної навчальної дисципліни. А також розробка корисного додатку, який зможе принести користь людям, отримання практичних навичок проектування та розробки додатку.

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

На сьогоднішній день люди все більше і більше часу вимушені проводити за роботою та побутовими клопотами, тому часу іноді просто не вистачає. Таким чином данна система повинна вирішувати цю проблему частково чи повністю.

Наразі існує деяка кількість аналогів даної системи. Оскільки проблема досить поширена попит на подібні системи не спадає. Але всі існуючі системи не прижилися серед населення у зв'язку з деякими факторами.

Метою є створення конкурентоспроможної системи обміну послугами, що дозволить скоротити втрати часу.

Потреби користувача включають:

- Можливість розміщення замовлень;
- Можливість підписатися на замовлення;
- Перевірка виконання замовлення.

У подальшому проект може бути розширеним та модифікованим згідно з подальшими потребами користувачів.

Архітектура системи спроектована для досягнення легкості масштабування та оптимізації навантажень на сервер.

Основні функції:

- отримання всіх замовлень для яких не призначений виконавч;
- отримання всіх замовлень по тегам користувача;
- підписка на замовлення;
- призначення одного з підписаних користувачей виконавцем замовлення;
- авторизація;
- реєстрація;
- перегляд профілю користувача.

Програмний продукт, повинен включати базу даних, яка зберігає дані про замовлення, користувачів, коментарії, теги та сервер, що обробляє запити.

Сервер повинен реалізовувати наступні функції:

- надавати інформацію для клієнтів;
- реєструвати запит від клієнта;
- обробляти запит від клієнта;
- зберігати отриману інформацію.

Він повинен бути реалізований за допомогою наступних фреймворків:

- Spring MVC;
- Spring MongoDB Framework.

На сервері повинна стояти MongoDB база даних, головною функцією якої є збереження даних, та отримання даних за необхідністю.

База даних буде зберігати наступні об'єкти:

- користувач;
- тег;
- замовлення;
- коментарій до замовлення;
- коментарій до користувача;
- роль.

Користувач буде мати такі поля як:

- id(унікальний номер користувача);
- ім'я користувача;
- пароль;
- ФІО;
- дата реєстрації;
- дата народження;
- рейтинг;
- баланс;
- роль.

Тег буде мати такі поля як:

- id;
- назва.

Замовлення буде мати такі поля як:

- id;
- ім'я;
- опис
- вартість;
- дата створення;
- теги;
- автор;
- чи підтверджене збоку автора;

- чи підтверджене з боку виконача;
- можливі виконачи;
- датчик.

Коментар до замовлення буде мати такі поля як:

- id;
- текст;
- дата створення;
- автор;
- замовлення.

Коментар до користувача буде мати такі поля як:

- id;
- текст;
- дата створення;
- автор;
- користувач.

Веб-сайт повинен реалізовувати наступні функції:

- відображення замовлень;
- відображення профілю користувача;
- підписки на замовлення;
- редагування бази даних через веб-сайт.

Веб сайт повинен бути реалізований за допомогою наступних технологій:

- Java;
- Spring Framework;
- Spring MVC;
- css - для реалізації дизайну сайту;
- Apache Tomcat - веб сервер;
- Spring Data для роботи з MongoDB базою даних.

Клієнтський додаток для мобільних телефонів на платформі Android для сповіщення повинен реалізовувати наступні функції:

- авторизація;
- реєстрація;
- відображення профілю;
- відображення замовлень.

Додаток повинен бути реалізований з використанням наступних технологій:

- Java;
- XML.

Варто зазначити, що додаток взаємодіє з базою даних MongoDB через HTTP запити до сервера. Тому додаток має всі ті ж класи моделі, що і сервер(користувач, тег, замовлення і т.д.).[2-3] Android - операційна система для смартфонів, інтернет-планшетів, електронних книг, цифрових програвачів, наручних годинників, ігрових приставок, нетбуків, смартбуків, окулярів Google, телевізорів та інших пристроїв. Заснована на ядрі Лінукс і власної реалізації віртуальної машини Java від Google. Спочатку розроблялася компанією Android, Inc., яку потім купила Google. Згодом Google ініціювала створення альянсу Open Handset Alliance (ОНА), який зараз займається підтримкою і подальшим розвитком платформи. Android дозволяє створювати Java-додатки, що керують пристроєм через розроблені Google бібліотеки. Android Native Development Kit[4] дозволяє перенести бібліотеки і компоненти додатків, написані на Сі та інших мовах.

У 86% смартфонів, проданих у другому кварталі 2014 року, була встановлена операційна система Андроїд. При цьому за весь 2014 рік було продано більше 1 мільярда Android-пристроїв. Саме через її популярність була і обрана саме ця платформа.

```
graph TD
    User((User))
    Admin((Administrator))
    UC1([View list of all orders])
    UC2([View list of all orders with filters])
    UC3([Manage account])
    UC4([Create order])
    UC5([Edit account])
    UC6([Delete account])
    UC7([Add comentary])
    UC8([Delete commentary])
    UC9([Change count])
    UC10([Change name])
    UC11([Change description])
    UC12([Manage user list])
    UC13([Delete user])
    UC14([Ban User])
    UC15([Unban User])
    UC16([Manage order list])
    UC17([Delete order from list])
    UC18([Delete commentary from order])
    UC19([Manage order])
    UC20([Change order])
    UC21([Create order])
    UC22([Delete order])

    User --> UC1
    User --> UC2
    User --> UC3
    User --> UC4
    Admin --> UC12
    Admin --> UC13
    Admin --> UC14
    Admin --> UC15
    Admin --> UC16
    Admin --> UC17
    Admin --> UC18
    Admin --> UC19
    Admin --> UC20
    Admin --> UC21
    Admin --> UC22

    UC2 -.-> UC1
    UC5 -.-> UC3
    UC6 -.-> UC3
    UC7 -.-> UC20
    UC8 -.-> UC20
    UC9 -.-> UC20
    UC10 -.-> UC20
    UC11 -.-> UC20
    UC13 -.-> UC12
    UC14 -.-> UC12
    UC15 -.-> UC12
    UC17 -.-> UC16
    UC18 -.-> UC16
    UC21 -.-> UC19
    UC22 -.-> UC19
    UC20 -.-> UC19
```

Рисунок 2.2 зображує діаграму класів. Ця діаграма демонструє відношення статичних, декларативних елементів моделі. Такий тип діаграм може застосовуватися як при розробці

нової системи, так і при зворотньому проектуванні. На даній діаграмі відображені основна моделі. Використані при розробці програмної системи.

Оскільки система є інформаційною, то діаграма класів є розгорнутою у відповідності до схеми баз даних.

Основними класами є Користувачі, Замовлення, Роль. Також є додаткові класи Репозиторіїв.

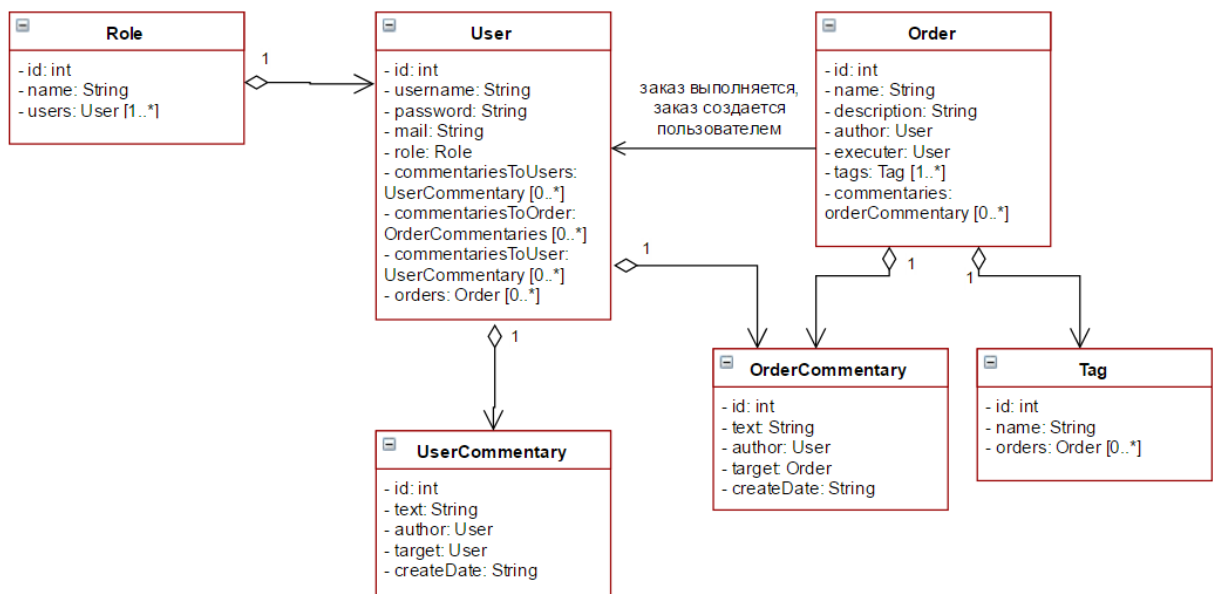


Рисунок 2.2 – Діаграма класів

Вона є однією з форм статичного опису системи з точки зору її проектування, показуючи її структуру. Діаграма класів не відображує динамічну поведінку об'єктів зображених на ній класів. На діаграмах класів показуються класи, інтерфейси і відносини між ними.

Клас - це основний будівельний блок. Це поняття є і в об'єктно-орієнтованих мовах програмування, тобто між класами UML і програмними класами є відповідність, що є основою для автоматичної генерації програмних кодів або для виконання реінжинірингу. Кожен клас має назву, атрибути і операції. Клас на діаграмі показується у вигляді прямокутника, розділеного на 3 області. У верхній міститься назва класу, в середній - опис атрибутів (властивостей), в нижній - назви операцій - послуг, що надаються об'єктами цього класу.

Діаграма станів (рис. 2.3) демонструє процес переходу станів замовлення при використанні користувачем. На даній діаграмі зображено стани замовлення.

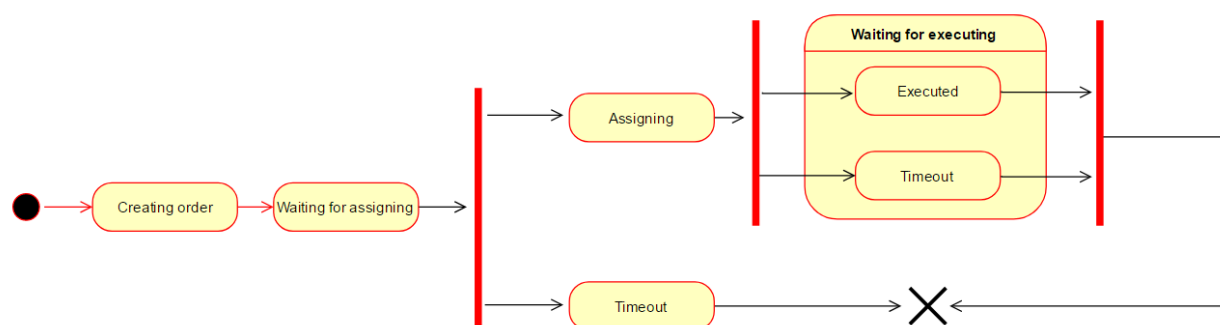


Рисунок 2.3 – Діаграма станів

Діаграма станів показує, як об'єкт переходить з одного стану в інший. Очевидно, що діаграми станів служать для моделювання динамічних аспектів системи (як і діаграми послідовностей, кооперації, прецедентів і, як ми побачимо далі, діаграми діяльності). Часто можна почути, що діаграма станів показує автомат, але про це ми поговоримо докладніше трохи пізніше. Діаграма станів корисна при моделюванні життєвого циклу об'єкта (як і її приватна різновид - діаграма діяльності, про яку ми будемо говорити далі).

На рисунку 2.4 зображена діаграма компонентів. Діаграма компонентів відображає залежності між компонентами користувацького інтерфейсу клієнтської частини додатку.

Проілюстрована вложеність компонентів серверу.

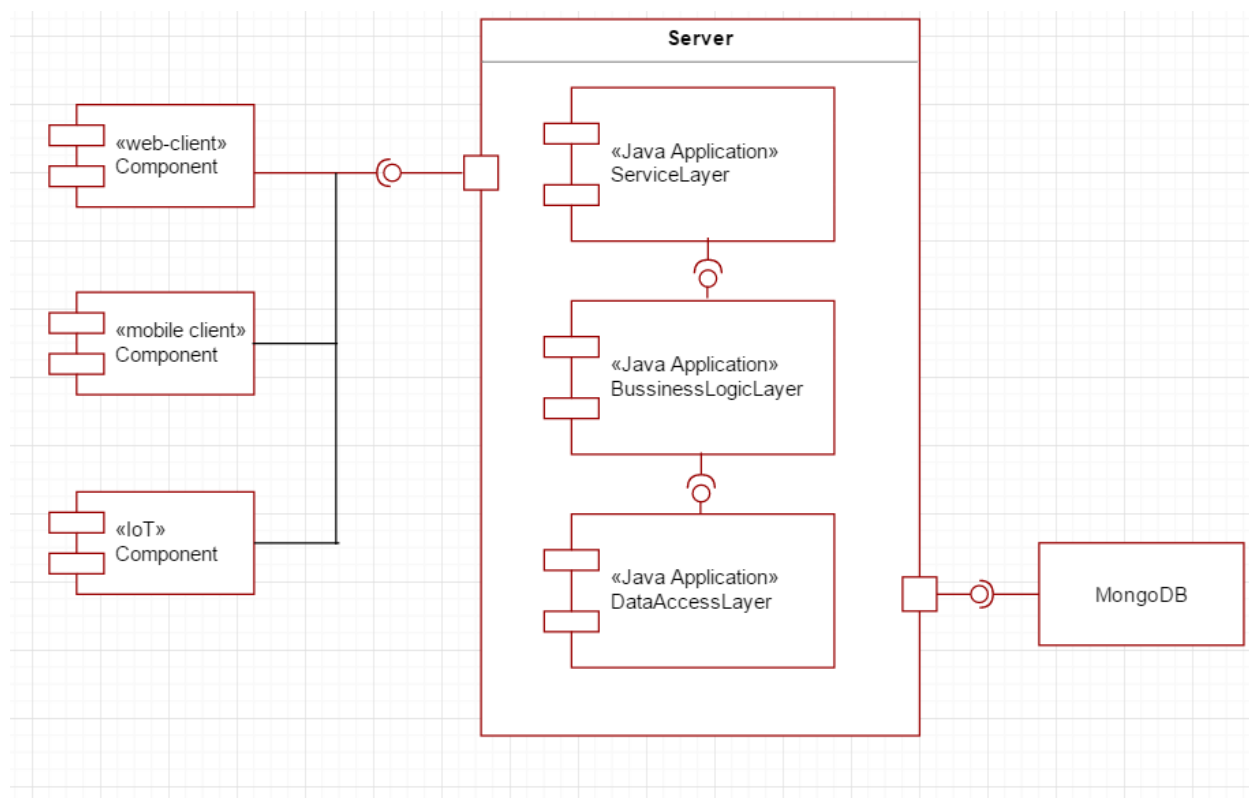


Рисунок 2.4 – Діаграма компонентів

Тепер, маючи діаграму компонентів, ми можемо побудувати діаграму розгортання для завершальної фази проекту. Вона зображена на рисунку 2.5.

На даній діаграмі зображується розміщення компонентів системи: на сервері додатку, сервері баз даних або мобільному додатку.

Фактично, діаграма розгортання повторює діаграму компонентів, за винятком того, що вона пояснює, які вузли працюють з якими програмними елементами. В іншому ж, вона також демонструє зв'язок вузлів між собою.

Варто зазначити, що передача від клієнта, як веб-сайту так і мобільного додатку, відбуватиметься за протоколом HTTP.

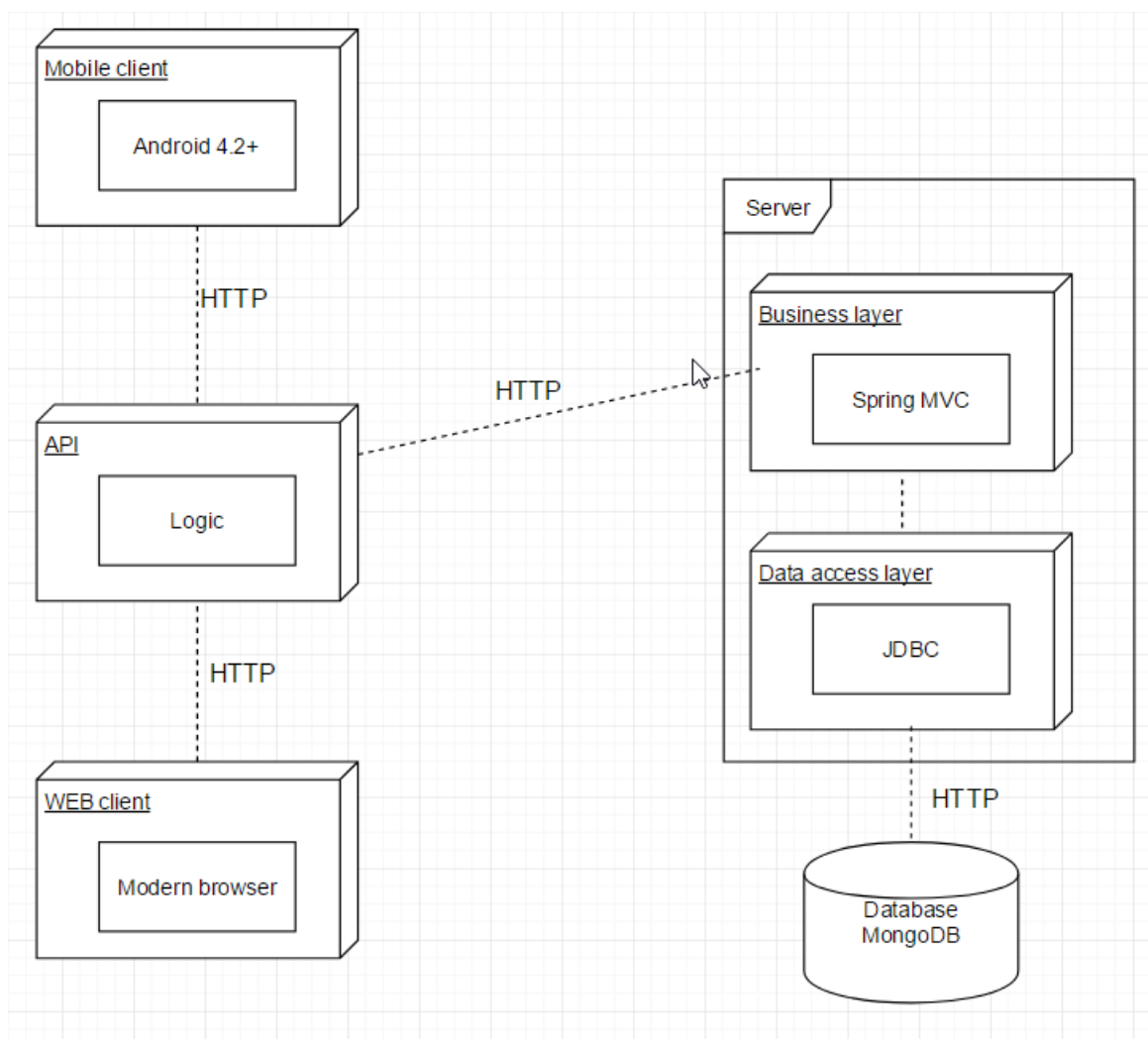


Рисунок 2.5 – Діаграма розгортання

Діаграма розгортання, Deployment diagram в UML моделює фізичний розгортання

артефактів на вузлах. Наприклад, щоб описати веб-сайт діаграма розгортання повинна показувати, які апаратні компоненти («вузли») існують (наприклад, веб-сервер, сервер бази даних, сервер додатки), які програмні компоненти («артефакти») працюють на кожному вузлі (наприклад, веб-додаток, база даних), і як різні частини цього комплексу з'єднуються один з одним (наприклад, JDBC, REST, RMI).

Загальна взаємодія об'єктами добре демонструється на прикладі діаграми послідовностей. На рисунку 2.6 зображено приклад даної діаграми для типової задачі системи, а саме розміщення замовлення. Для цього користувач повинен відкрити веб-сторінку розміщення замовлення, ввести необхідні данні(назву, опис, ціну та теги) та натиснути кнопку «Розмістити». При цьому клієнт відправляє на сервер запит, сервер отримує замовлення та додає його до бази даних.

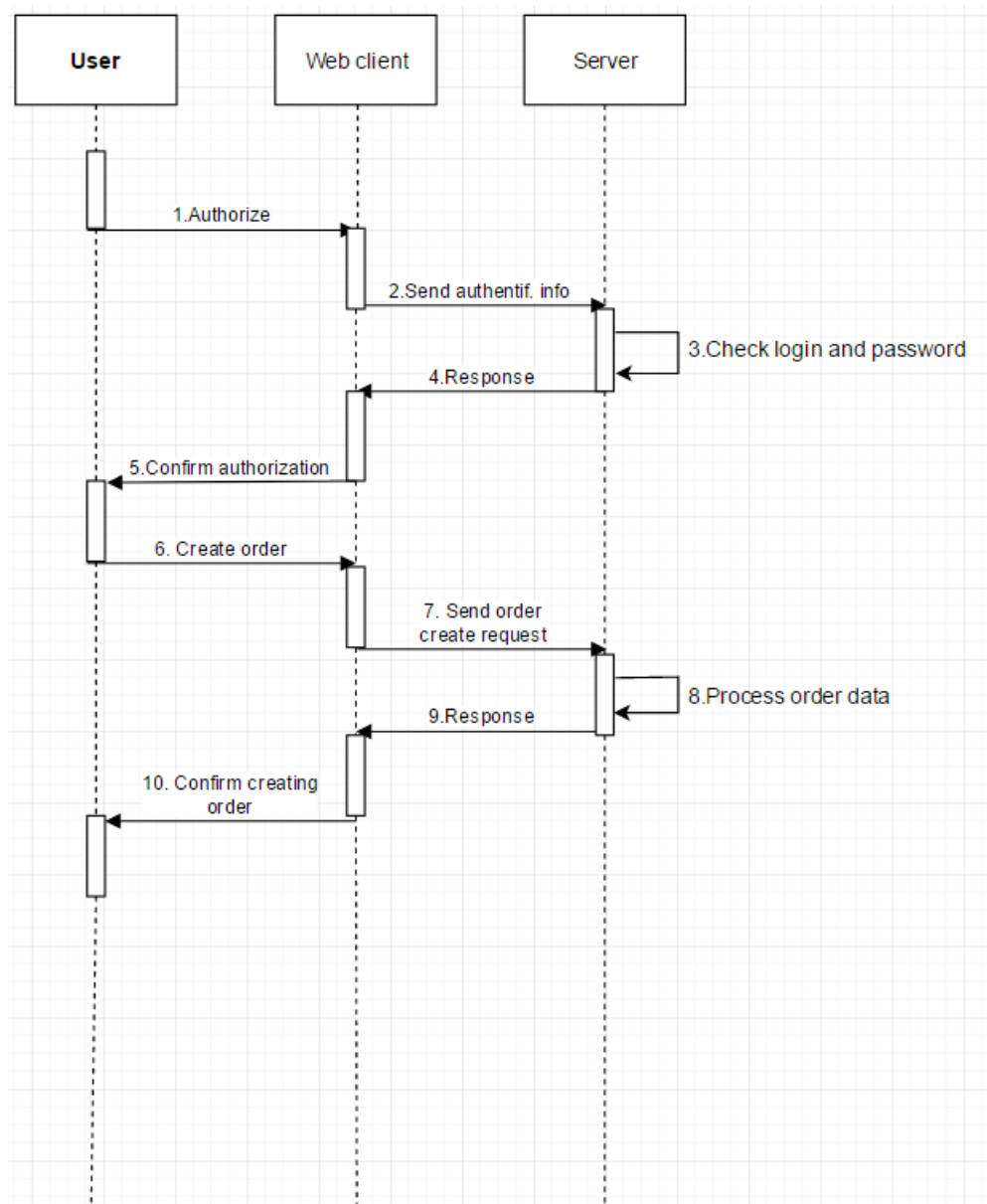


Рисунок 2.6 – Діаграма послідовностей

Діаграма послідовності (англ. sequence diagram) - діаграма, на якій для деякого набору об'єктів на єдиній тимчасовій осі показаний життєвий цикл будь-якого певного об'єкта (створення-діяльність-знищення якоїсь сутності) і взаємодія акторів (дійових осіб) ІС в рамках якого- або певного прецеденту (відправка запитів і отримання відповідей). Використовується в мові UML.

Основними елементами діаграми послідовності є позначення об'єктів (прямокутники з назвами об'єктів), вертикальні «лінії життя» (англ. Lifeline), що відображають плин часу, прямокутники, що відображають діяльність об'єкта або виконання ним певної функції (прямокутники на пунктирною «лінії життя»), і стрілки, що показують обмін сигналами або повідомленнями між об'єктами.

На даній діаграмі об'єкти розташовуються зліва направо.

Після діаграми послідовностей варто зобразити діаграму кооперації, оскільки вони уявляють собою схожі схеми, як за семантикою, так і за структурою (див. рис. 2.7).

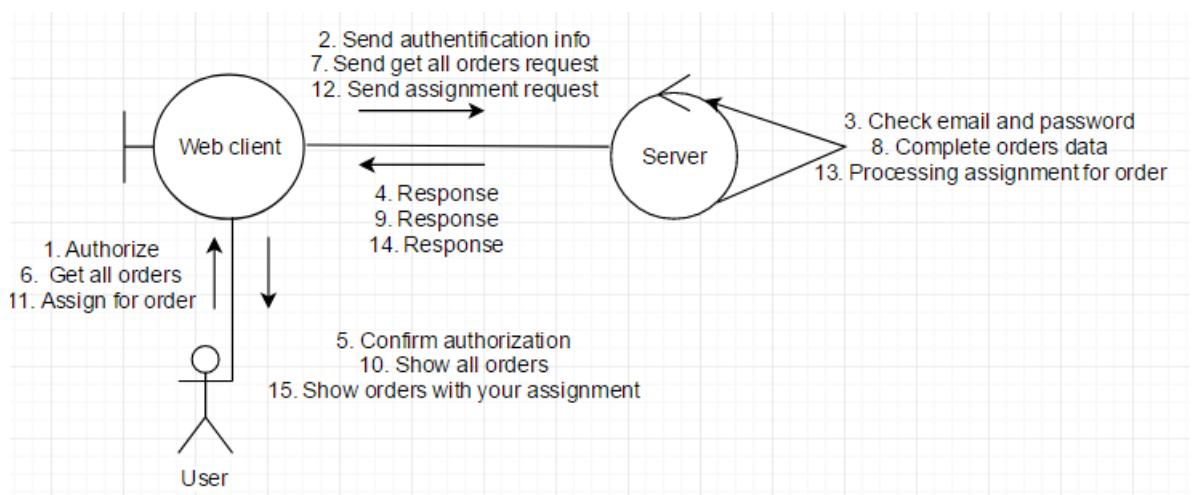


Рисунок 2.7 – Діаграма кооперації

На даній діаграмі зображена також типова задача, але тут більший акцент зроблено власне на об'єкти, що приймають участь у взаємодії. Головна особливість діаграми кооперації полягає в можливості графічно представити не тільки послідовність взаємодії, але і всі структурні відносини між об'єктами, які беруть участь в цій взаємодії.

Перш за все, на діаграмі кооперації у вигляді прямокутників зображуються беруть участь у взаємодії об'єкти, що містять ім'я об'єкту, його клас і, можливо, значення атрибутів. Далі, як і на діаграмі класів, вказуються асоціації між об'єктами у вигляді різних сполучних ліній. При цьому можна явно вказати імена асоціації і ролей, які грають об'єкти в даній асоціації. Додатково можуть бути зображені динамічні зв'язки - потоки повідомлень. Вони представляються також у вигляді сполучних ліній між об'єктами, над

якими розташовується стрілка з вказівкою напрямку, імені повідомлення і порядкового номера в загальній послідовності ініціалізації повідомлень.

На відміну від діаграми послідовності, на діаграмі кооперації зображаються тільки відносини між об'єктами, що грають певні ролі у взаємодії. На цій діаграмі не вказується час у вигляді окремого виміру. Тому послідовність взаємодій і паралельних потоків може бути визначена за допомогою порядкових номерів. Отже, якщо необхідно явно специфікувати взаємозв'язки між об'єктами в реальному часі, краще це робити на діаграмі послідовності.

Діаграма активності (рисунок 2.8) яка демонструє основну активність користувача в системі.

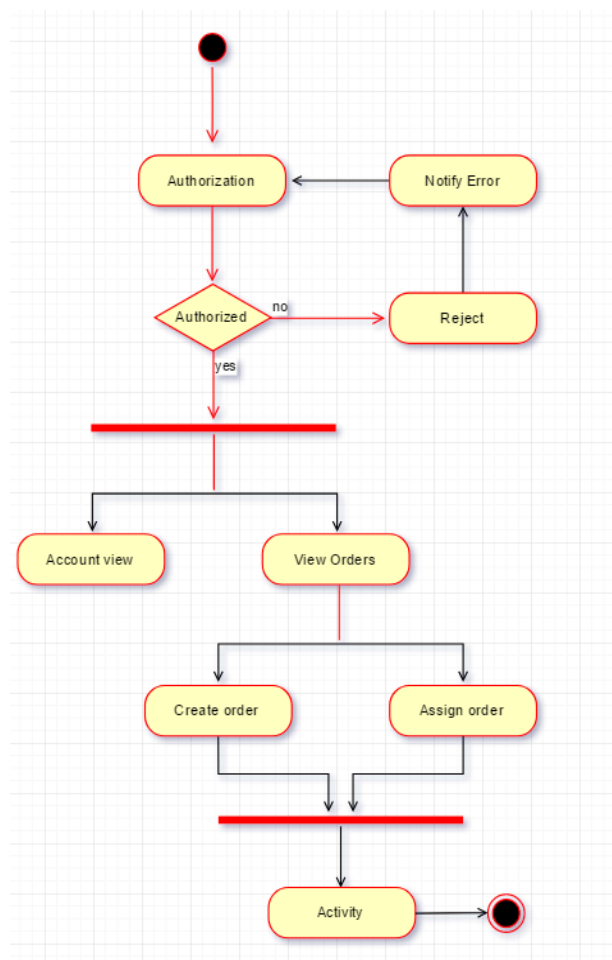


Рисунок 2.8 – Діаграма активності

Після авторизації можливі варіанти отримання доступу або відмови в доступі. Надалі діяльність системи та зміна її станів залежить лише від користувача.

На рисунку 2.9 зображено діграму об'єктів, яка демонструє лише частку, що стосується моделей системи, не торкаючись сутностей, що використовуються додатково.

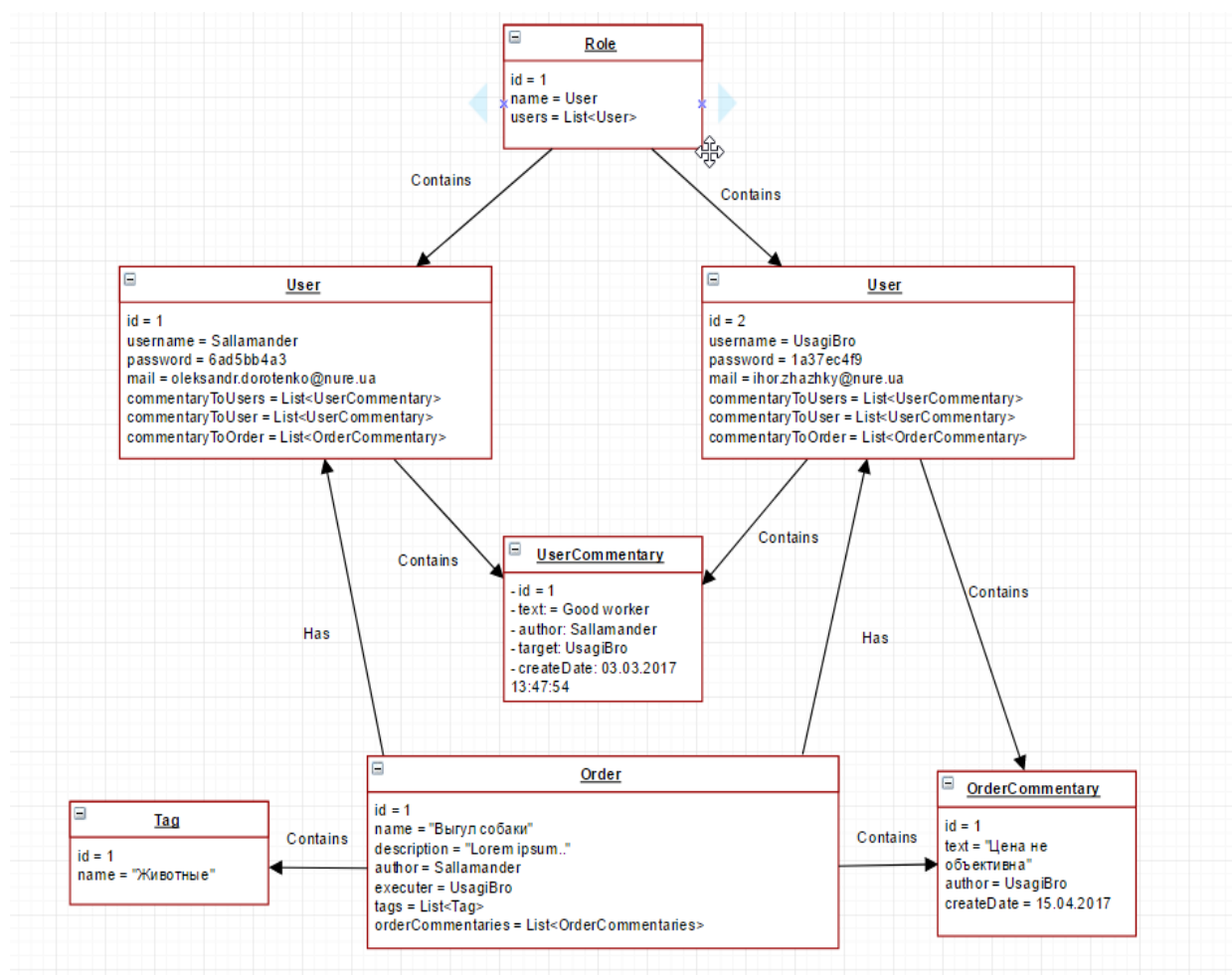


Рисунок 2.9 – Діаграма об'єктів

Дана діаграма є логічним продовженням діаграми класів з відображенням відповідних об'єктів та відношень між ними.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

Під час планування та моделювання програмного продукту було визначено, що проект має складатися з чотирьох частин:

- веб-сайт для створення та редагування замовлення;
- сервер з базою даних, на якому будуть оброблятися запити ззовні і зберігатися вся інформація про систему та дані (сервер у свою чергу є трьохрівневим, проте логічно ці частини можна об'єднати у єдине поняття серверу, через тісну взаємодію між ними);
- мобільний клієнт, розроблений під Android, який повинен реалізувати більшу частину функціоналу веб-сайту, а саме: реєстрація, авторизація, перегляд замовлень, розміщення замовлення, підписка на замовлення та інше;
- ІОТ пристрій – емулятор датчику виконання замовлення, який буде розроблена на мові програмування Java. Його функція в тому, що він за допомогою спеціальних датчиків буде реєструвати виконання замовлення вдома у автора замовлення та автоматично присвоювати замовленню статус «Виконано».

3.1 Загальні відомості про систему

Під час моделювання і планування програмного продукту було встановлено, що проект повинен складатися з таких частин:

- веб сайт;
- сервер;
- база даних;
- мобільний додаток.
- IoT.

Вся система була побудована на Spring Framework. Spring Framework (або коротко Spring)[4-5] - універсальний фреймворк з відкритим вихідним кодом для Java-платформи. Spring забезпечує вирішення багатьох завдань, з якими стикаються Java-розробники і організації, які хочуть створити інформаційну систему, засновану на платформі Java. Через широку функціональність важко визначити найбільш значущі структурні елементи, з яких він складається. Spring не повністю пов'язаний з платформою Java Enterprise, незважаючи

на його масштабну інтеграцію з нею, що є важливою причиною його популярності.

Для веб-сайту був використаний Spring MVC[8], який побудований на мові програмування Java і розортається на серверах Apache Tomcat. Цей фреймворк був використаний мною, так як він дуже зручний у використанні, та має зручні інтерфейси для роботи з HTTP протоколами. Метою SpringMVC Framework є підтримка у Spring архітектури модель-уявлення-контролер (model-view-controller). Spring забезпечує готові компоненти, які можуть бути використані (і використовуються) для розробки веб-додатків.

Головною метою MVC є поділ об'єктів, бізнес-логіки й зовнішнього вигляду програми. Всі ці компоненти слабо пов'язані між собою і при бажанні ми можемо змінити, наприклад, зовнішній вигляд програми, не вносячи суттєві зміни в інші два компонента. HTML, CSS – для зовнішнього оформлення і дизайну системи, також був використаний фреймворк Bootstrap. Для серверу був використаний також Spring MVC, та бібліотека GSON для роботи с JSON форматом від Google. Як база даних була обрана MongoDB, так як вона нереляційна, тому значно швидша за реляційні аналоги. Мобільний додаток був написаний на мові програмування Java, з використанням Android SDK. Варто зазначити, що коректна робота додатку гарантується на версії Android 4.4.2 та вище. В якості Internet of Things пристрою виступав емулятор написаний також на мові програмування Java.

3.2 Серверна частина

Серверна частина є центральною у контексті системи, на ній знаходиться уся бізнес логіка, підключення до бази даних та інше[6-7].

Сервер розроблено багаторівневим (де на кожному фізичному рівні виконується певний логічний «шар» програми, тобто шар сервісів, бізнес-логіки та доступу до даних) з метою забезпечення ефективного масштабування системи, зокрема вертикальної, та більших можливостей для подальшої оптимізації.

API кожного рівня представляє собою пакет контролерів, які відповідають на зовнішні запити.

Контролери ініціалізуються під час запуску url, через що є постійно доступними, та забезпечують можливість обробки значної кількості запитів від користувачів. Обмін даними, як між частинами серверу, так і між мобільним, веб додатком, датчиками і сервером відбувається з використанням HTTP-запитів, при чому дані передаються у форматі JSON.

3.3 Мобільний клієнт

Останній з перерахованих вище компонентів системи і один з важливих її складових – це мобільний клієнт.

Мобільний додаток було реалізовано з використанням Android SDK під ОС Android. Він отримує дані з сервера за рахунок HTTP-запитів. HTTP запити обробляються класом RestTemplate який реалізований у додатку до фреймворка Spring MVC.

3.4 Веб сайт

Для першої версії проекту цілком достатньо мати простий веб-сайт, який дозволить користувачам створювати замовлення. Сайт дає відвідувачам змогу ознайомитися з певною інформацією.

Головні технології, використані для розробки веб-сайту були вказані вище (усі вони зосереджені навколо Java).

На даний момент веб-сайт підтримує такий функціонал:

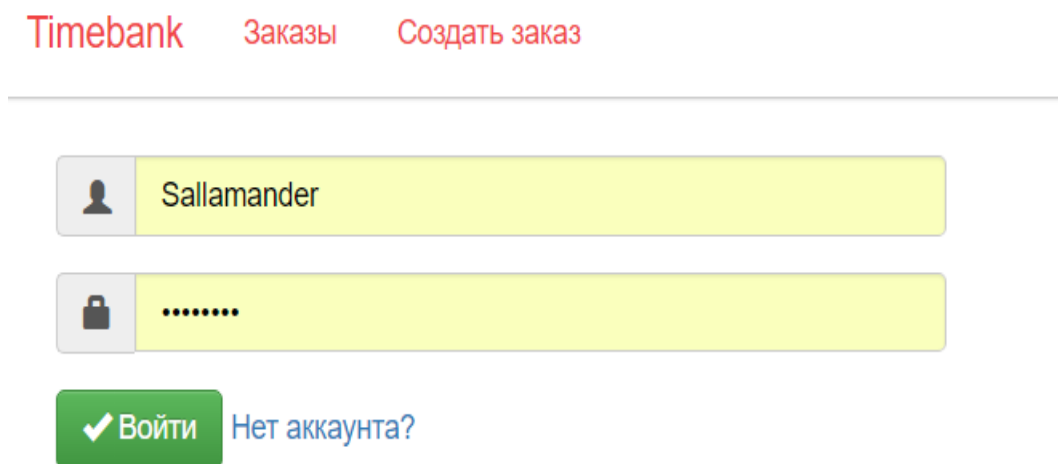
- авторизація;
- реєстрація;
- створення замовлення;
- зміна паролю;
- зміна інформації про себе;
- зміна обраних тегів;
- підписка на замовлення;
- вибір виконача замовлення;
- перегляд профілю користувача;
- підтвердження виконання замовлення.

4 ІНТЕРФЕЙС І ФУНКЦІОНАЛ

Система складається з чотирьох компонентів, тому розробка проводилась окремо по етапам. Варто зауважити, що такі компоненти як сервер та API закриті для користувача.

4.1 Веб сайт

Для того, щоб розпочати роботу з системою, потрібно зайти на головну сторінку додатку. Неавторизований користувач може перейти до системи, але жодний функціонал не буде доступний для використання через обмеження прав доступу. Тому система переадресує на сторінку авторизації (див. рис. 4.1), де користувач має ввести свій персональний логін та пароль. Лише після авторизації користувач зможе повноцінно використовувати функціонал системи.

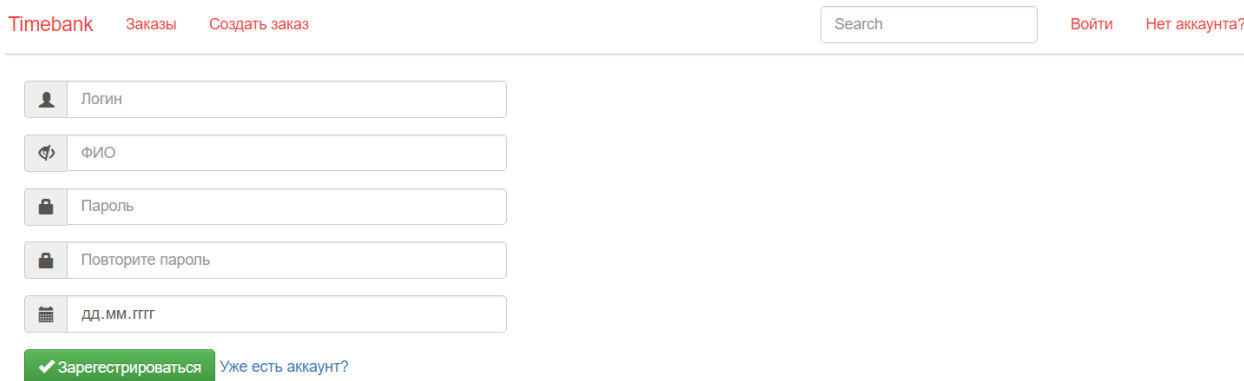


Timebank Заказы Создать заказ

[Нет аккаунта?](#)

Рисунок 4.1 — Авторизація у системі

Якщо користувач не має аккаунта у системі він може зареєструватися (рисунок 4.2).



Timebank Заказы Создать заказ

Search Войти Нет аккаунта?

Логин

ФИО

Пароль

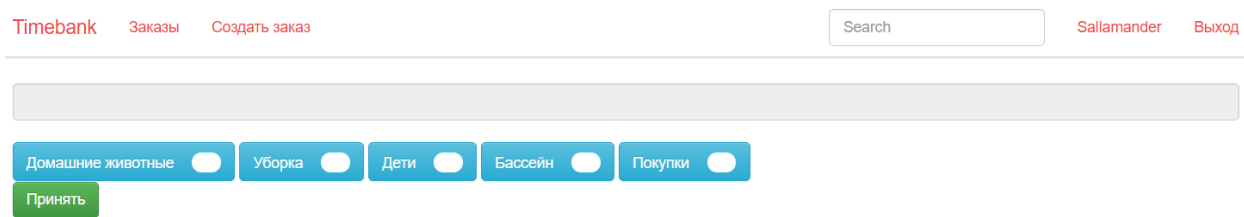
Повторите пароль

дд.мм.гггг

✓ Зарегистрироваться Уже есть аккаунт?

Рисунок 4.2 — Вікно реєстрації у системі

Після цього користувачу буде запропоновано обрати теги які його цікавлять, для того, щоб зробити зручну фільтрацію замовлень(див. рис. 4.3).



Timebank Заказы Создать заказ

Search Sallamander Выход

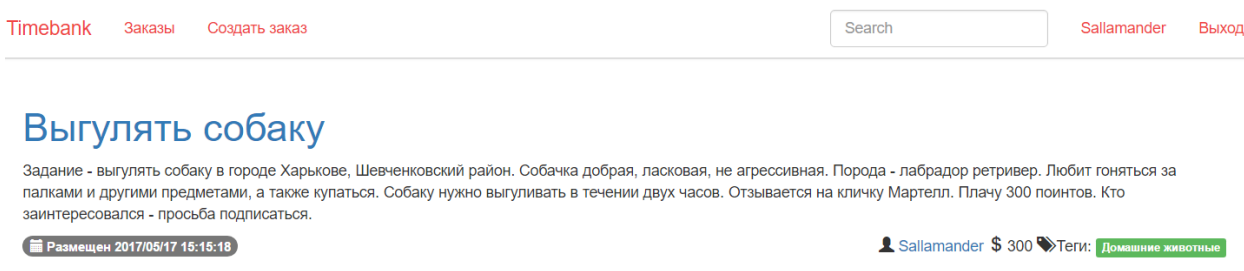
Домашние животные ☐ Уборка ☐ Дети ☐ Бассейн ☐ Покупки ☐

Принять

Рисунок 4.3 — Вибір тегів

На даний момент в системі присутні лише п'ять тегів, але адміністратор завжди може додати ще.

Користувач одразу потрапляє на сторінку з замовленнями (рисунок 4.4).



Timebank Заказы Создать заказ

Search Sallamander Выход

Выгулять собаку

Задание - выгулять собаку в городе Харькове, Шевченковский район. Собачка добрая, ласковая, не агрессивная. Порода - лабрадор ретривер. Любит гоняться за палками и другими предметами, а также купаться. Собаку нужно выгуливать в течении двух часов. Отзывается на кличку Мартелл. Плачу 300 поинтов. Кто заинтересовался - просьба подписаться.

Размещен 2017/05/17 15:15:18

Sallamander \$ 300 Тег: Домашние животные

Рисунок 4.4 – Список усіх замовлень

Тут він може відкрити замовлення або створити замовлення. Після натиснення на назву замовлення він потрапляє на сторінку замовлення (рисунок 4.5).

Выгулять собаку

Задание - выгулять собаку в городе Харькове, Шевченковский район. Собачка добрая, ласковая, не агрессивная. Порода - лабрадор ретривер. Любит гоняться за палками и другими предметами, а также купаться. Собаку нужно выгуливать в течении двух часов. Отзывается на кличку Мартелл. Плачу 300 поинтов. Кто заинтересовался - просьба подписаться.

Размещен 2017/05/17 15:15:18

Sallamander \$ 300 Теги: Домашние животные

Комментарии:

UsagiBro

Слишком занижена цена для такого задания

Размещен: 2017/05/17 15:15:18

Пожаловаться?

Рисунок 4.5 – Сторінка замовлення

Автор замовлення на цій сторінці може призначити виконача замовлення (рисунок 4.6).

Выгулять собаку

Задание - выгулять собаку в городе Харькове, Шевченковский район. Собачка добрая, ласковая, не агрессивная. Порода - лабрадор ретривер. Любит гоняться за палками и другими предметами, а также купаться. Собаку нужно выгуливать в течении двух часов. Отзывается на кличку Мартелл. Плачу 300 поинтов. Кто заинтересовался - просьба подписаться.

Размещен 2017/05/17 15:15:18

Sallamander \$ 300 Теги: Домашние животные

Управление заказом:

Выберите пользователя для выполнения:

UsagiBro

Назначить

Комментарии:

UsagiBro

Слишком занижена цена для такого задания

Размещен: 2017/05/17 15:15:18

Пожаловаться?

Рисунок 4.6 – Сторінка призначення виконача на замовлення

Далі якщо виконач успішно виконав замовлення ви можете підтвердити це (рисунок 4.7). Після цього сума яка є ціною замовлення додається до його балансу.

Timebank

Заказы

Создать заказ

Search

Sallamander

Выход

Личные данные:

Логин:

Sallamander

ФИО:

Dorotenko Aleksandr Vadimovich

Дата рождения:

03.07.1997

Дата регистрации:

2017/05/17

Баланс:

999899

Изменить данные

Изменить пароль

Изменить теги

Теги на которые вы подписаны:

Вы не подписались ни на один тег. Хотите сделать это сейчас?

Заказы, которые разместили вы:

Выгулять собаку

Покрасить окна

Подтвердить выполнение заказа?

Личные данные:

Логин:

Sallam

ФИО:

Doroten

Дата рождения:

03.07.1

Дата регистрации:

2017/0

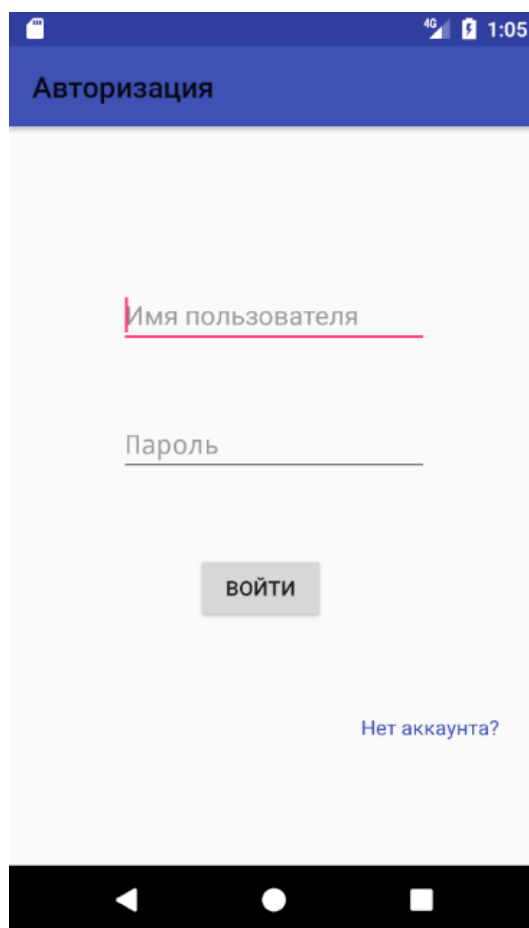
Баланс:

999899

Рисунок 4.7 – Підтвердження виконання замовлення з боку автора

4.2 Мобільний клієнт

При відкритті додатку користувачу пропонується ввести його логін та пароль або зареєструватися (див. рис. 4.8).



The screenshot shows a mobile application interface for user authentication. At the top, there is a blue header bar with the title 'Авторизация' (Authorization) in white text. Below the header, the main area is light gray. It contains two input fields: the first is labeled 'Имя пользователя' (Username) and has a red vertical line on its left side; the second is labeled 'Пароль' (Password). Below these fields is a gray button with the text 'ВОЙТИ' (Login). At the bottom right of the form area, there is a blue link that says 'Нет аккаунта?' (No account?). The bottom of the screen shows a black navigation bar with three white icons: a back arrow, a circle, and a square.

Рис. 4.8 – Форма авторизації

Після успішної авторизації користувач одразу потрапляє до сторінки свого профілю з якої він має змогу відкрити список замовлень, або вийти з аккаунту, що зображено на рисунку 4.10.

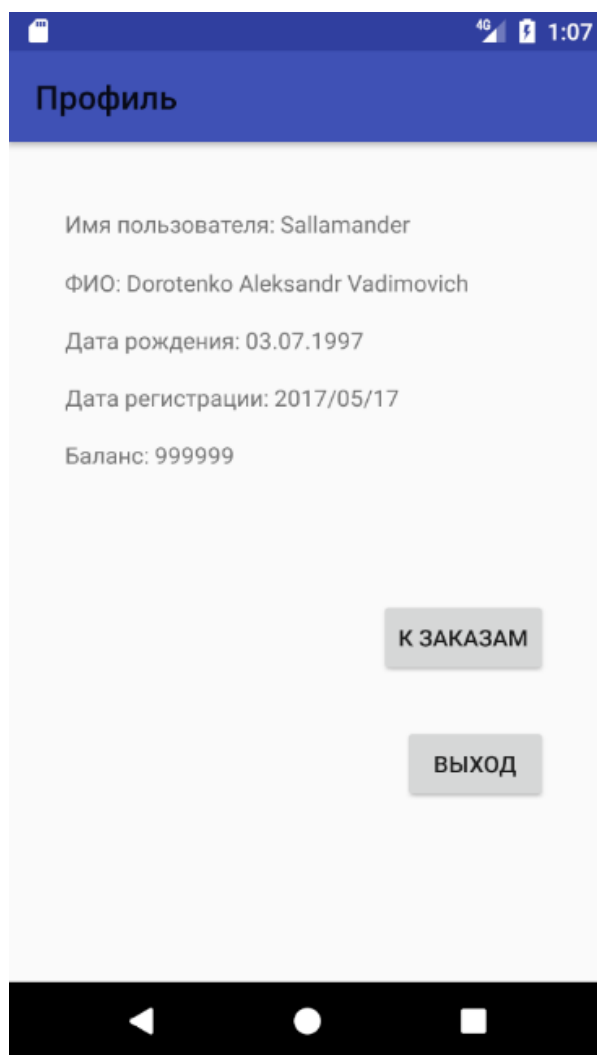


Рис. 4.10 – Пункты меню

Далі користувач може натиснути кнопку «К заказам» та потрапити на сторінку розміщення всіх замовлень. Де за допомогою елементу «Recycler View» будуть послідовно розміщені ім'я та опис кожного замовлення (рис. 4.11).

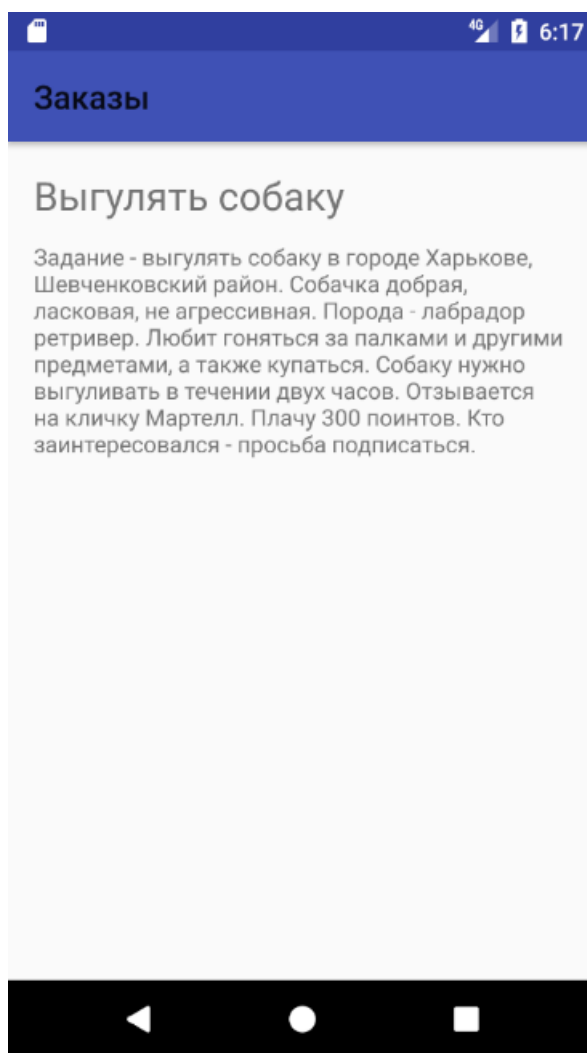


Рисунок 4.11 — Сторінка всіх замовлень

Мобільний додаток взаємодіє на пряму з сервером, відправляючи йому HTTP запити. Сервер в свою чергу відправляє запит до бази даних та отримавши відповідь відправляє відповідь до мобільного додатку у форматі JSON. JSON - текстовий формат обміну даними, заснований на JavaScript. Як і багато інших текстові формати, JSON легко читається людьми. Формат JSON був розроблений Дугласом Крокфордом.

Незважаючи на походження від JavaScript (точніше, від підмножини мови стандарту ECMA-262 1999 роки), формат вважається незалежним від мови і може використовуватися практично з будь-якою мовою програмування. Для багатьох мов існує готовий код для створення і обробки даних в форматі JSON. Я використовував бібліотеку Gson від Google для опрацювання цього формату. Бібліотека GSON була розроблена програмістами Google. Сама вона написана на Java і дозволяє конвертувати об'єкти JSON в Java-об'єкти і навпаки. Також у Android додатку був використаний XML для створення представлень та конфігурування. XML - розширювана мова розмітки. Рекомендований Консорціумом

Всесвітньої павутини (W3C).[9] Специфікація XML описує XML-документи і частково описує поведінку XML-процесорів (програм, які читають XML-документи і забезпечують доступ до їх вмісту). XML розроблявся як мова з простим формальним синтаксисом, зручний для створення і обробки документів програмами і одночасно зручний для читання і створення документів людиною, з підкресленням націленості на використання в Інтернеті.

ВИСНОВКИ

В результаті виконання курсової роботи було створено веб-сервіс для обміну послугами «TimeBank» за допомогою технології Spring MVC[10], розробка клієнтського додатка з елементами адміністрування за допомогою мови програмування Java.

Результатом курсового проекту є система, що складається з клієнту та серверу, яка дозволяє користувачам обмінюватися послугами. Система дозволяє користувачам розміщувати замовлення та виконувати їх.

Для реалізації всього необхідного функціоналу була проаналізована предметна область, виявлені взаємовідносини між основними об'єктами системи. Для зберігання інформації була спроектована база даних, яка відображає всі сутності даної предметної області.

Дана інформаційна система є багатофункціональною у використанні, має доступний інтерфейс, що дозволяє працювати з програмою користувачам з різним рівнем комп'ютерної грамотності, має середні вимоги до апаратного та програмного забезпечення, забезпечує досить високу швидкість роботи, безпечна у використанні, що є явними плюсами цього додатка. Система максимально проста та зрозуміла.

Отримана система дозволяє користувачам виконувати такі дії:

- Перегляд списку замовлень;
- Розміщення замовлення;
- Підписка на замовлення;
- Сортування замовлень по тегах;
- Підтвердження виконання замовлень;
- Зміна обраних тегів;

При виконанні курсової роботи я вирішив проблеми обміну даними між клієнтом, сервером та IoT.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Блох Д. Effective Java / Джошуа Блох. – Львів: Абабаламага, 2008. – 259 с. .
2. Брюс Эккель. Философия Java. — 3 вид. — Спб.: Питер, 2003. — 976 с.
3. Барри Берд. Java 8 для чайников — М.: «Диалектика», 2015. — 400 с.
4. Фрэд Лонг, Дхрув Мохиндра, Роберт С. Сикорд, Дин Ф. Сазерленд, Дэвид Свобода. Руководство для программиста на Java: 75 рекомендаций по написанию надежных и защищённых программ. — М.: «Вильямс», 2014. — 256 с.
5. Кей С. Хорстманн, Гари Корнелл. Java. Библиотека профессионала, том 1. Основы. 9-е издание. — М.: «Вильямс», 2013. — 864 с.
6. Фаулер М. Рефакторинг[Текст] / Мартин Фаулер. — Київ: Знання, 2015. — 378 с.
7. Фаулер М. Шаблоны корпоративных приложений, Мартин Фаулер. — М.: «Вильямс», 2012. — 544 с.
8. Кен Арнольд, Джеймс Гослинг Мова програмування Java. — Пітер: Пітер-Пресс, 1996. 205с.
9. Дэвид Хантер, Джефф Рафтер, Джо Фаусетт, Эрик ван дер Влиет, и др. XML. Работа с XML, 4-е издание = Beginning XML, 4th Edition. — Москва: «Диалектика», 2009. — 1344 с.
10. Кларенс Хо, Роб Харроп. Spring 3 для профессионалов. — Москва.: «Вильямс», 2012. — 880 с.