МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

БІЛЯЧАТ ДМИТРО ІВАНОВИЧ

УДК 004.51

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ БІРЖИ ФРІЛАНСУ З ВИКОРИСТАННЯМ OLAP ТЕХНОЛОГІЙ

Автореферат кваліфікаційної роботи на здобуття

ступеня вищої освіти «Бакалавр»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня кваліфікація

«Бакалавр з інженерії програмного забезпечення»

Миколаїв – 2021

Кваліфікаційною роботою є рукопис.

Робота виконана в Чорноморському національному університеті імені Петра Могили Міністерства освіти і науки України на кафедрі інженерії програмного забезпечення.

Керівник: д-р техн. наук,
професор
Фісун Микола Тихонович

Рецензент: д-р техн. наук,
професор
Кондратенко Галина Володимирівна

Захист відбудеться «22» червня 2021 р. о 9 год. на засіданні екзаменаційної комісії (ауд. 2-309) у Чорноморському національному університеті імені Петра Могили за адресою: вул. 68 Десантників, 10, Миколаїв, 54003.

З кваліфікаційною роботою можна ознайомитися в бібліотеці Чорноморського національного університету імені Петра Могили за адресою: вул. 68 Десантників, 10, Миколаїв, 54003.

Автореферат представлений «__»_____2021 р.

Секретар
екзаменаційної комісії,
викладач кафедри ІПЗ Кандиба Ігор Олександрович

Загальна характеристика роботи

Актуальність: в сучасному світі все більше уваги приділяється вебзастосункам. Все більше компанії переносять свої застосунки в Інтернет, наприклад: MS Office. Дана реалізація застосунків надає, практично, той самий досвід використання, що й прикладні програми.

Розглядаючи таку предметну область, як фріланс біржа, неможливо обійтись без інформаційної системи, так як саме ця система надає найбільш зручний спосіб взаємодії користувача і системи. Найбільшою перевагою даного рішення є можливість доступу до застосунку з будь-якої операційної системи за умови наявності Інтернету.

Вебзастосунок - розподілений застосунок, в якому клієнтом виступає браузер, а сервером - вебсервер. Браузер може бути реалізацією так званих тонких клієнтів, у яких логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у відображенні інформації, завантаженої мережею з сервера, і передачі даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-застосунки є міжплатформеними сервісами

Актуальність створення системи для фріланс біржі – це доступ до послуг з будь-якого пристрою, засоби підтримки даних в актуальному стані, а також миттєве оновлення.

Об’єкт дослідження – вебтехнології для фріланс біржі.

Предмет дослідження – вебзатосунок на основі трирівневої архітектури з функіоналом обробки статистичних даних.

Мета роботи: поліпшення комунікації між замовником та виконавцем проєктів за допомогою трирівневої архітектури вебзастосунку та удосконалення блока статистичного аналізу.

Для досягнення зазначеної мети поставлено такі **завдання:**

- аналіз предметної сфери та існуючих аналогів;
- розробка структури бази даних;

- аналіз сучасних методів реалізації веб-застосунків;
- розробка веб-застосунку для фріланс біржі;
- опрацювання способів обробки статистичних даних, отриманих за допомогою технологій OLAP;
- тестування застосунку.

КРБ викладено на 77 сторінках, вона містить 4 розділи, 41 ілюстрацію, 1 таблицю, 17 джерел в переліку посилань.

Ключові слова: вебpfcnjceyjr, біржа, проектування, OLAP, база даних, фріланс, трирівнева архітектура, сервер, користувач.

У першому розділі кваліфікаійної роботи бакалавра доводиться актуальність обраної теми. Представлено основні види веб-застосунків та проведено порівняльний аналіз сучасних аналогів системи, що розробляється, та пошук їх недоліків. Сформовано основні вимоги до програмного забезпечення розроблюваного вебзастосунку.

Основні вимоги до функіональності:

- реєстрація користувачів;
- авторизація користувачів;
- реалізація архітектури 3tier web application;
- можливість додавати свої технічні завдання, для замовників;
- можливість взяти на виконання певний проект, для виконавців;
- можливість додати власне резюме;
- відображення коментарів по проектам та рейтингу користувачів;
- можливість додати опис компанії;
- реалізація зручного мессенжеру;
- зручна організація бази даних;
- реалізація сучасних технологій для розробки ПЗ;
- зручна панель адміністратора;
- зручний та багатофункціональний інтерфейс;
- велика кількість сценаріїв використання;
- обробка помилок;

- особистий кабінет для операторів;
- наявність декількох ролів користувачів;
- зручний користувацький інтерфейс системи;

Вимоги до функцій роботи користувача:

Користувачі:

- замовник. Виставляє технічне завдання проекту, який треба виконати та обирає спеціаліста для реалізації;
- виконавець (фрілансер) – виконує технічне завдання;
- адміністратор – додає контент до сайту та слідкує за працездатністю;
- агент сервісу підтримки.Опис структури системи:

Система буде мати клієнт-серверну структуру, де в якості клієнту буде виступати браузер користувача, в якості серверу – застосунок на фреймворку Django, а також сервер баз даних. Клієнтська та серверна частина будуть взаємодіяти через API (рис. 1).

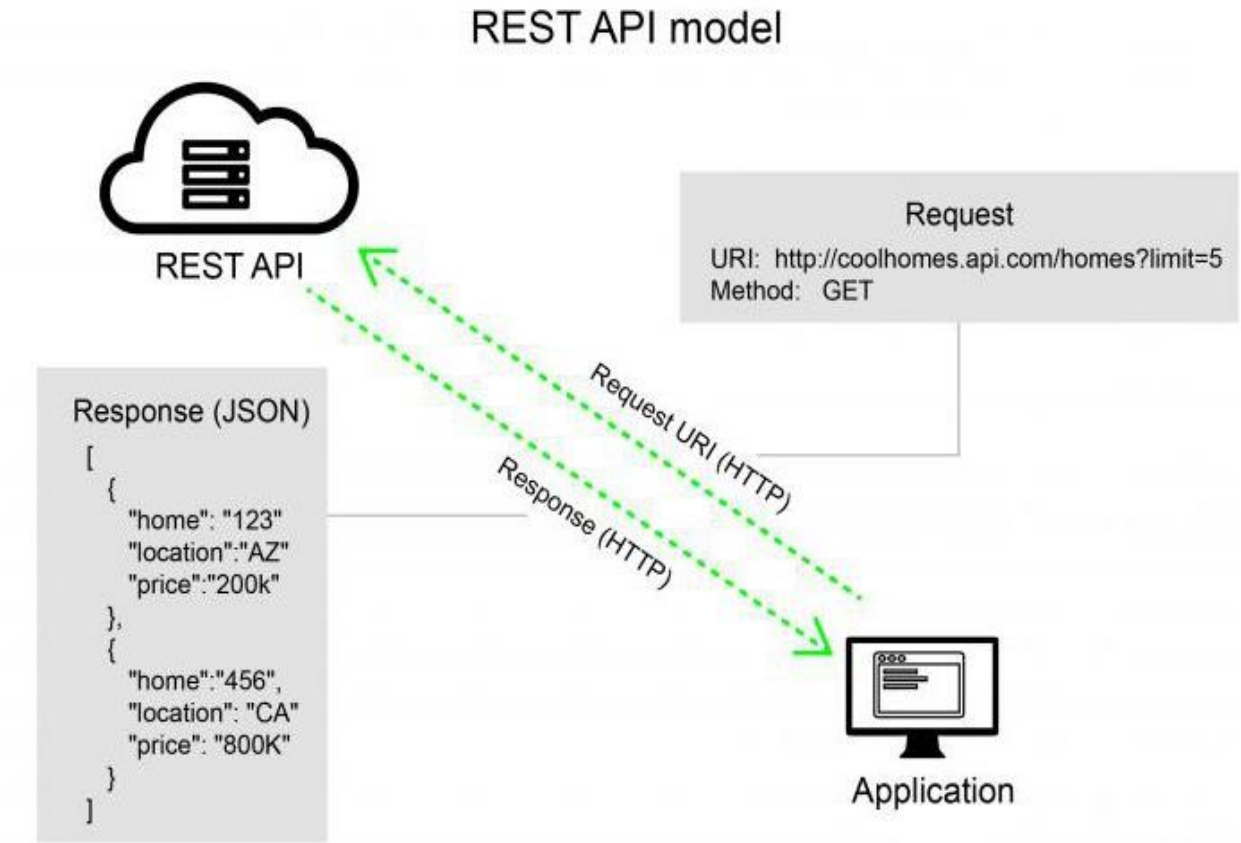

Рисунок 1 – схема структури застосунку

Сценарії роботи системи:

- Користувач (замовник) заходить на сторінку реєстрації, в спеціальні поля вводить необхідні дані, які відправляються та обробляються сервером. Потім у кабінеті користувача додає проект, який потрібно виконати та через деякий час обирає виконавця із списку фріланцерів, що відгукнулись.
- Виконавець реєструється, заходить на сторінку відображення всіх замовлень, обирає потрібний. У нього є можливість зв’язатися з замовником для уточнення деталей. Виконує роботу та відправляє замовнику через систему платежів.
- Система безпечних платежів отримує дані замовника та виконавця, переводить кошти.
- Замовник заходить на сторінку фрілансера та залишає йому відгук та ставить відповідну оцінку (1 - 5).
- Агент підтримки авторизується в системі, читає надіслані повідомлення, вирішує проблеми та робить звіт.

Засоби програмної реалізації:

- фреймворк Python Django;
- фреймворк Vue.js;
- Microsoft SQL Server;
- браузер.

Засоби апаратної реалізації:

- сервер;
- персональний комп’ютер.

У другому розділі описано виконану роботу з моделювання та конструювання ПЗ. Розроблено архітектуру ПЗ на основі обраних компонентів.

Для розробки вебзастосунку обрано клієнт-серверну архітектуру.

Клієнт-серверна архітектура це така архітектура (спосіб організації структури вебзастосунку), в якій завдання, поставлені перед системою, умовно діляться між двома підсистемами:

- клієнтом;
- сервером;

При цьому важливо розуміти, що:

- Сервер - є "центральною" частиною, зберігає дані і до нього звертається безліч клієнтів. Можна сказати, що сервер може жити без клієнтів, а от клієнти без сервера не можуть.
- Клієнт - програма, примірників якої зазвичай "більше", ніж серверів. Наприклад, до одного веб-сервера (що зберігає сайт) може підключатися велике число веб-браузерів. Зазвичай саме з програмою-клієнтом працюють користувачі (тому її так і називають), з іншого боку клієнтом сервера може бути і робот.
- У дуже навантажених застосунках серверів теж може бути дуже багато (але це означає часто все одно означає, що клієнтів у рази більше)

Система має клієнт-серверну структуру, де в якості клієнту буде виступати браузер користувача, в якості серверу – застосунок на фреймворку Django, а також сервер баз даних. Клієнтська та серверна частини будуть взаємодіяти через API .

Як і будь-яка інша технологія, клієнт-серверна архітектура має свої переваги і свої недоліки. Розглянемо їх.

Переваги: виконання більшої частини роботи потужною серверною частиною при мінімумі навантаження на клієнта, основна частина даних зберігаються на сервері. При цьому, як правило, він краще захищений від різного виду загроз, ніж звичайний клієнтський ПК, можливість більш чіткого розмежування повноважень доступу до різних рівнів інформаційної системи (кожному клієнту - свій рівень доступу), кросплатформеність, простіше кажучи, будь-який клієнт може працювати з ресурсами сервера незалежно від операційної системи, зменшення навантаження на мережу з

огляду на те, що клієнт в основному передає сервера команди, а той вже їх виконує.

Недоліки: вихід з ладу сервера може привести до непрацездатності всієї системи, висока вартість серверного обладнання та його обслуговування (зокрема, може знадобитися окремий фахівець для обслуговування), високе навантаження на серверне обладнання і канал зв'язку до нього.

У третьому розділі представлено виконану роботу з кодування, тестування розробленого програмного забезпечення, проведено тестування, створенно керівництво користувача.

При розробці Python-додатків або використанні рішень на Python, створених іншими розробниками, може виникнути ряд проблем, пов'язаних з використанням бібліотек різних версій.

Для вирішення даних питань використовується підхід, заснований на побудові віртуальних оточень - свого роду пісочниць, в рамках яких запускається додаток зі своїми бібліотеками, оновлення та зміна яких не торкнеться інших додаток, що використовують ті ж бібліотеки.

При розробці застосунку створено віртуальне оточення за допомогою вбудованого в Python інструмент «venv».

Для реалізації клієнтської частини створено та налаштовано новий застосунок за допомогою інструментів Vue CLI (рис. 2).

```
kantegory@kantegory:~/channel/naive-freelance/django-freelance$ cd ..
kantegory@kantegory:~/channel/naive-freelance$ vue init webpack vue-freelance

? Project name vue-freelance
? Project description Naive freelance frontend
? Author kantegory <davidobryakov@gmail.com>
? Vue build standalone
? Install vue-router? Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Airbnb
? Set up unit tests No
? Setup e2e tests with Nightwatch? No
? Should we run `npm install` for you after the project has been created? (recommended) npm

   vue-cli · Generated "vue-freelance".


# Installing project dependencies ...
# ========================
```

Рисунок 2 – створення нового проєкту

Vue CLI - повноцінна система для швидкої розробки на Vue.js. Vue CLI

прагне стати стандартним інструментарієм для екосистеми Vue. Він забезпечує безперебійну роботу різних інструментів збірки, встановлює розумні значення за замовчуванням, тому ви зможете зосередитися на розробці програми, а не проводити дні за його налаштуванням. У той же час, залишається сучасною функціональністю конфігурації кожного інструменту без необхідності вилучення конфігурації в окремий файл.

Наступним кроком стало налаштування та запуск локальних серверів, на яких працюють серверна (Django) та клієнтська (Vue.js) частина.

Серверна частина вебзастосунку - це програма або скрипт на сервері, що обробляють запити користувача (точніше, запити браузера). При кожному переході користувача по посиланню браузер відправляє запит до сервера. Сервер обробляє цей запит, викликаючи деякий Python-скрипт, який формує вебсторінку, або структуру даних (JSON), і відсилає клієнтові по мережі. Браузер тут же відображає отриманий результат у вигляді чергової веб-сторінки.

Застосунок на Python Django складається з 3 основних частин (файлів) – views.py, models.py, urls.py.

View, або уявлення, - це те місце, де розміщено «логіку» роботи застосунку. Воно запросить інформацію з моделі, створеної раніше, і передасть її в клієнтську частину, яка вже буде відображати і дані.
Для передачі даних до клієнтської частини, було конвертовано їх до формату JSON за допомогою сериалайзерів.

У четвертому розділі виконано роботу з конструювання та створення OLAP кубу для данних розробленого застосунку. Представлено основні переваги та недоліки використання OLAP.

В основі концепції OLAP лежить куб OLAP. OLAP-куб - це структура даних, оптимізована для дуже швидкого аналізу даних.

Куб OLAP складається з числових фактів, які класифікуються за вимірюваннями. OLAP Cube також називають гіперкубом (рис. 3).

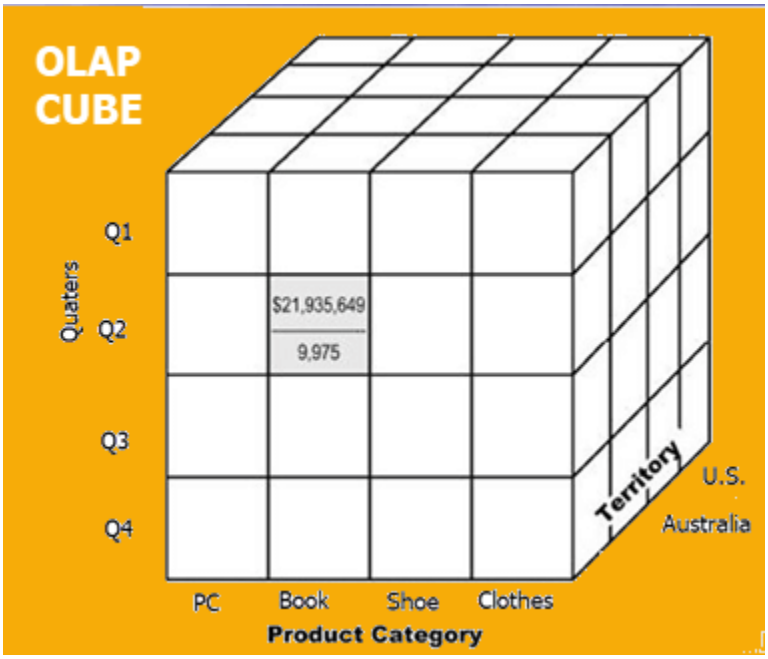

Рисунок 3–OLAP куб

Зазвичай операції з даними і аналіз виконуються з використанням простої електронної таблиці, де значення даних розташовуються в форматі рядків і стовпців. Це ідеально підходить для двовимірних даних. Однак OLAP містить багатовимірні дані, причому дані зазвичай отримують з іншого і незв'язаного джерела. Використання електронної таблиці не є оптимальним варіантом. Куб може зберігати і аналізувати багатовимірні дані в логічній і упорядкованій формі.

Сховище даних буде отримувати інформацію з декількох джерел даних і форматів, таких як текстові файли, таблиці Excel, мультимедійні файли і т. Д.

Витягнуті дані очищаються і перетворюються. Дані завантажуються на сервер OLAP (або куб OLAP), де інформація попередньо розраховується заздалегідь для подальшого аналізу.

Факт - це числова величина яка розташовується в осередках гіперкуба.

Вимірювання (dimension) - це безліч об'єктів одного або декількох типів, організованих у вигляді ієрархічної структури і забезпечують інформаційний контекст числового показника. Вимірювання прийнято візуалізувати у вигляді ребра багатовимірного куба.

Об'єкти, сукупність яких і утворює вимір, називаються членами

вимірювань (members). Члени вимірювань візуалізують як точки або долі, що відкладаються на осях гиперкуба.

Комірка (cell) - атомарна структура куба, відповідна повного набору конкретний значень вимірів.

Ієрархія - групування об'єктів одного виміру в об'єкти більш високого рівня. Наприклад - день-місяць-рік. Ієрархії в вимірах необхідні для можливості агрегації і деталізації значень показників відповідно до їх ієрархічній структурі. Ієрархія цілком грунтується на одному вимірі і формується з рівнів.

В OLAP-системах підтримуються наступні базові операції:

- поворот;
- проекція. При проекції значення в осередках, що лежать на осі проекції, підсумовуються по деякому зумовленої законом;
- розкриття (drill-down). Одне зі значень вимірювання замінюється сукупністю значень з наступного рівня ієрархії виміру; відповідно замінюються значення в осередках гіперкуба;
- згортка (roll-up / drill-up). Операція, зворотна розкриттю;
- перетин (slice-and-dice).

Теоретичні дослідження проводились на основі аналізу існуючих аналогів, актуальності використаних технологій та технічної літератури.

Висновки

Під час виконання кваліфікаційної роботи бакалавра було проведено аналітичну роботу по способам розробки сучасних вебзастосунків, пошуку аналогічних систем, класифікації вебсайтів. Побудовано UML-діаграми для проєктування архітектури системи. Обрано сучасні інструменти для розробки користувацької і серверної частини.

Створено веб-застосунок для роботи фріланс біржи. Система дозволяє працювати з даними пов’язаними з контентом та адмініструванням користувачів у додатку.

Аналізуючи виконану роботу, можна стверджувати, що задачі, які були поставленні на початку роботи, були досягнуті. Первинним завданням був аналіз предметної області, що дозволяло зрозуміти необхідні напрями роботи, зрозуміти логіку проекту. Першочерговими задачами були виділення та автоматизація процесів та функцій. Технічне завдання передбачало основні функції, які повинна виконувати система. Успішне виконання усієї системи ґрунтується на правильності розробленої структури застосунку, а також його раціонального проектування.

Перед створенням програмного забезпечення було проаналізовано існуючі аналогічні системи, що були створені раніше. Були виділені позитивні та негативні властивості, якими володіють системи.

На етапі моделювання були створені різноманітні моделі систем та створено дизайн веб-застосунку та UML діаграми різних типів.

Забезпечено базу даних та виконання різних операції з даними, які в ній зберігаються. Використано СКБД для підтримки цілісності даних, забезпечення незалежності даних від програм, централізоване зберігання інформації та надання спільного доступу до даних кільком клієнтам. За допомогою Microsoft Analytics Services створено OLAP куб, це дозволило зручно аналізувати дані та керувати робочим процесом. Проведено тестування системи, що дозволяє впевнитись в її стабільності.

Було вжито такі технології: Microsoft SQL Server (реляційна СКБД), Python (інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією), Django (фреймворк реалізований на Python), HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), JavaScript (скриптова мова для веб-сторінок) та Vue.js (фреймворк для розробки користувацького інтерфейсу).