МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №4**

з дисципліни «Веб-технології та веб-дизайн-2.»

Варіант №13

**Виконав:**

студент гр. БС-83

Розмариця О.А.

**Перевірив:**

ас. Давидько О. Б.,

ас. Матвійчук О. В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2020

**Мета роботи:**

Ознайомитися із створенням API та роботою з даними на основі обраних раніше фреймворків.

**Завдання:**

Створити простий додаток, що реалізує частину функціоналу, згідно архітектури, що була розроблена у попередній роботі. Вимоги до реалізації:

1. Реалізувати аутентифікацію та авторизацію
2. Створити точки доступу для базових CRUD операцій.
3. У випадку моноліту мають бути реалізовані операції для двох модулів та підключений Redis як кеш-сервер
4. Для мікросервісів мають бути реалізовані операції для одного з них + авторизація, як окремий сервіс, та API Gateway.

Дозволяється використання будь-яких фреймворків на обраній мові.

**Контрольні питання:**

1. Різниця між аутентифікацією та авторизацією

Аутентифікація використовується для підтвердження особи зареєстрованого користувача. Перевірка справжності - це процес перевірки облікових даних: ідентифікатора користувача (імені, адреси електронної пошти, номери телефону) і пароля. Якщо ідентифікатор і пароль співпадають із записами, що зберігаються в базі даних системи, користувачеві надається доступ.

Авторизація відбувається після того, як особистість користувача успішно аутентифицирующей системою. Процес авторизації визначає, чи має минулий перевірку осіб доступ до певних ресурсів: інформації, файлів, бази даних.

1. Що таке Redis?

Redis (Remote dictionary server) - резидентна система управління базами даних класу NoSQL з відкритим вихідним кодом, що працює зі структурами даних типу «ключ - значення». Використовується як для баз даних, так і для реалізації кешей, брокерів повідомлень.

Орієнтована на досягнення максимальної продуктивності на атомарних операціях

1. Алгоритми витіснення даних із кеша.

**Алгоритм Беладі** - відкидати з кешу ту інформацію, яка не знадобиться в майбутньому найдовше.

**Least recently used (LRU)**: в першу чергу, витісняються значення, які не використовувались найдовше. Цей алгоритм вимагає відстеження того, що і коли використовувалося, що може виявитися досить накладно, особливо якщо потрібно проводити додаткову перевірку, щоб в цьому переконатися.

**Most Recently Used (MRU):** на відміну від LRU, в першу чергу витісняється останній використаний елемент.

**Псевдо-LRU (PLRU)**: Для кешей з великою асоціативністю (зазвичай більше 4 каналів), необхідні ресурси для реалізації LRU стають занадто великими. Якщо достатня політика, що майже завжди потрібно відкидати найменш використовуваний елемент, то в цьому випадку можна використовувати алгоритм PLRU, що вимагає для елемента кешу тільки один біт.

**Сегментований LRU** (Segmented LRU або SLRU): «SLRU-кеш ділиться на два сегменти. пробний сегмент і захищений сегмент. Рядки в кожному сегменті впорядковані від часто використовуваних до найменш використовуваним. Дані при промахах додаються в кеш, причому в область останніх використаних елементів пробного сегмента. Дані при влучань прибираються де б вони не розташовувалися і додаються в область часто використовуваних елементів захищеного сегмента. До рядків захищеного сегмента звернення таким чином відбуваються принаймні двічі.

**Кеш прямого відображення**: для високошвидкісних кешей процесора, де не вистачає швидкодії 2-канального асоціативного кешування. Адреса нового елемента використовується для обчислення місцезнаходження в кеші (у відведеній для цього області). Все, що було раніше, - витісняється.

**Least Frequently Used (LFU):** LFU підраховує як часто використовується елемент. Ті елементи, звернення до яких відбуваються рідше за все, витісняються в першу чергу.

**Multi Queue Caching Algorithm**

Враховуються такі моменти:

* Елементи з різною вартістю: зберігання елементів, запит яких дуже дорогий, наприклад, такі, отримання яких потребує багато часу.
* Елементи, що вимагають більше місця в кеші: якщо елементи мають різний розмір, то контролер кеша може спробувати витіснити більший елемент, щоб зберегти кілька елементів поменше.
* Елементи, що застарівають з плином часу: Деякі кеші зберігають застарілу інформацію (наприклад, кеш новин, DNS-кеш або кеш веб-браузера). Комп'ютер може витіснити елементи внаслідок їх старіння. Залежно від розміру кеша, кешування нових елементів може зажадати витіснення старих.

Існують також різні алгоритми для забезпечення когерентності кеша. Це застосовується у випадках тільки тоді, коли безліч незалежних кешей використовується для зберігання однієї і тієї ж інформації (наприклад, безліч серверів баз даних оновлюють загальний файл даних).

1. HTTP методи для CRUD операцій.

POST, GET, PUT и DELETE

1. Що таке ORM?

ORM (Object-Relational Mapping) - технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних». Існують як пропрієтарні, так і вільні реалізації цієї технології.

1. Anemic та Rich моделі опису сутностей в ООП. Переваги та недоліки кожної з моделей.

Переваги Anemic:

1. Простота побудови. Це великий плюс, бо головне завдання архітектора, реалізація функціональності за менші гроші. 90 відсотків рішень, можуть бути побудовані в даній моделі і це буде на порядок дешевше.

2. Бізнес-об'єкти - отчуждаеми. В результаті, бізнес-об'єкт може бути спокійно пронесена від DAL, до фасаду і навіть далі. Беспрепятсвенно і ідентично серіалізовані / десеріалізован між фізичними шарами. У більшості, бізнес-об'єкт як бізнес-сутність не змінюється при перенесенні між шарами.

3. Прогнозованість і керованість аж до DAL. Що під цим розуміється. У разі, якщо ви ворочаєте великими обсягами даних, в Anemic вам простіше управляти запитами до бази даних, ніж в Rich. База даних була і залишається найважчою частиною бізнес додатків. Оптимізація в основному досягається за рахунок підвищення ефективності роботи з базою даних (і часто не звичайним редагування SQL). У Anemic, ви спокійно можете провести той, чи інший сценарій через сусідній сервіс, який буде працювати саме над цьому важким сценарієм. В Rich - проблема як з прогнозом результуючого запиту, і з його оптимізацією.

4. Бізнес-об'єкт простіше управляти об'єктами, які ще не повністю в валідованого стані. В Rich - наявність валідаторів зобов'язує тримати валідованого стан. Що іноді виливається в проблеми (наприклад, коли об'єкт тільки що створений, не має ідентифікатора, і не має тих, чи інших посилань). Багато в чому, бізнес-об'єкт більше схожий на дані, ніж на об'єкт в стилі об'єктного програмування.

Переваги Rich:

1. Простота використання. Об'єкти завжди під рукою. Засоби обробки об'єктів, також під рукою. Використовувати Rich - значно простіше, особливо коли в проекті новачок.

2. Інкапсуляція. Програмісту, що використовує такий об'єкт, недоступно його стан, крім як певний інтерфейс. Це локалізує деякі зміни в логіці. Але тут слід згадати, що ті зміни, які стосуються стану, також відслідковуються компіляцією із статичною типізацією в Anemic, що покриває більшу кількість таких проблем.

3. Менша кількість сміття. У Anemic доводиться відслідковувати, щоб в купі сервісів не робили свої велосипеди. В Rich c цим менше проблем в силу сильнішою локалізації логіки.

Завдання виконано і доступне за посиланням: https://github.com/AlexRull/Web