Задание на практическую работу

Цель

Целью выполнения практических заданий является получение практических навыков в создании приложений, ориентированных на хранение и обработку больших объемов данных.

Содержание курсовой работы

В рамках выполнения практической работы необходимо спроектировать и реализовать приложение, которое будет осуществлять хранение, обработку и предоставление доступа к данным, согласно варианту задания.
Обязательным требованием является применение следующих техник, обеспечивающих оптимальное управление данными:

1. При проектировании необходимо применить практики Architecture As A Code с использованием языка описания архитектуры C4 DSL.
2. Приложение должно осуществлять хранение данных о пользователях в реляционной базе данных. Должны быть построены индексы для эффективного поиска данных;
3. Приложение должно поддерживать кеширование данных о пользователях с помощью Redis, для ускорения запросов по ключевым полям;
4. Приложение должно осуществлять хранение данных задания в Document-ориентированной базе данных.
5. Реализация программного REST API для осуществления получения, изменения и поиска данных. Спецификация API должна быть представлена в Swagger;
6. Приложение должно осуществлять аутентификацию запросов к данным.
7. Программное обеспечение должно работать под управление операционной системы Linux Ubuntu Server версии 20 и выше на архитектуре x86/arch64
8. Развертывание приложения должно выполняться в контейнерах docker. (https://www.docker.com/). Должны быть отдельные контейнеры для каждого сервиса. Запуск контейнеров должен осуществляться с помощью утилиты docker-compose.
9. Необходимо создать скрипты для первоначальной инициализации СУБД и создание «начльных» данных.
10. Генерация нагрузки для проверки производительности REST API должны производиться с помощью WRK (https://github.com/wg/wrk)

Для разработанного программного обеспечения провести проверку нагрузки, на каждый из API.

Структура работы

Работа делается и сдается последовательно в виде следующих заданий:

Задание 01: Проектирование программной системы (Architecture As A Code)

Цель

Ознакомится с инструментами проектирования в формате Architecture As A Code.
Получить практический навык в моделировании в нотации C4

Задание

1. Установить инструменты из списка
    - Клиент Git
    - Текстовый редактор (рекомендуется Visual Studio Code)
    - Плагины к Visual Studio Code C4 DSL
2. Зарегистрироваться на github.com (если еще нет учетной записи)
3. Создать публичный репозиторий для выполнения практической работы у себя в аккаунте
4. Скопировать репозиторий https://github.com/DVDemon/hl_mai_lab_00 с примерами задания
5. Создать файлы с описанием «архитектуры» согласно вашему варианту задания в Structurizr Lite.
6. Требования к диаграммам:
    - Должна быть контекстная диаграмма
    - Должна быть диаграмма контейнеров
    - Должна быть диаграмма развертывания
    - Должно быть несколько динамических диаграмм

Задание 02: Stateful сервис для RDBMS

Цель

Получение практических навыков в построении сервисов, работающих с реляционными данными.

Задание

Разработать приложение осуществляющее хранение данных о пользователях в реляционной СУБД. Для выявленных в предыдущем задании вызовов между сервисами создайте REST интерфейс.

Должны выполняться следующие условия:

- Данные должны храниться в СУБД PostgreSQL;
- Должны быть созданы таблицы для каждой сущности из вашего задания;
- Интерфейс к сущностям должен предоставляться в соответствии со стилем REST;
- API должен быть специфицирован в OpenAPI 3.0 (должен хранится в index.yaml);
- Должен быть создан скрипт по созданию базы данных и таблиц, а также наполнению СУБД тестовыми значениями;
- Для сущности, отвечающей за хранение данных о пользователе (клиенте), для пользователей должен быть реализован интерфейс поиска по маске фамилии и имени, а также стандартные CRUD операции.
- Данные о пользователе должны включать логин и пароль. Пароль должен храниться в закрытом виде (хэширован)

Рекомендуема последовательность выполнения работы:

- Создайте схему БД
- Создайте таблицы
- Создайте скрипт для первичного наполнение БД и выполните
- Реализуйте REST-сервис
- Сделайте спецификацию с OpenAPI с помощью Postman и сохраните ее в index.yml
- Протестируйте сервис
- Создайте Dockerfile для вашего сервиса
- Протестируйте его работу в Docker
- Опубликуйте на github проект

Задание 03: Stateful сервис для NoSQL

Цель

Получение практических навыков в построении сервисов, работающих с документо-ориентированными базами данных.

Задание

Разработать приложение осуществляющее хранение данных согласно варианта задания в MongoDB. Для описанных в архитектуре вызовов между сервисами создайте REST интерфейс.

Должны выполняться следующие условия:

- Данные должны храниться в СУБД MongoDB;
- Интерфейс к сущностям должен предоставляться в соответствии со стилем REST;
- API должен быть специфицирован в OpenAPI 3.0 (должен хранится в index.yaml);
- Должен быть создан скрипт по созданию базы данных и таблиц, а также наполнению СУБД тестовыми значениями;
- Для каждой коллекции должен быть создан отдельный сервис, реализующий CRUD операции.

Рекомендуема последовательность выполнения работы:

- Создайте скрипт для первичного наполнение БД и выполните
- Реализуйте REST-сервис
- Сделайте спецификацию с OpenAPI с помощью Postman и сохраните ее в index.yml
- Протестируйте сервис
- Создайте Dockerfile для ваших сервисов
- Протестируйте его работу в Docker
- Опубликуйте на github проект

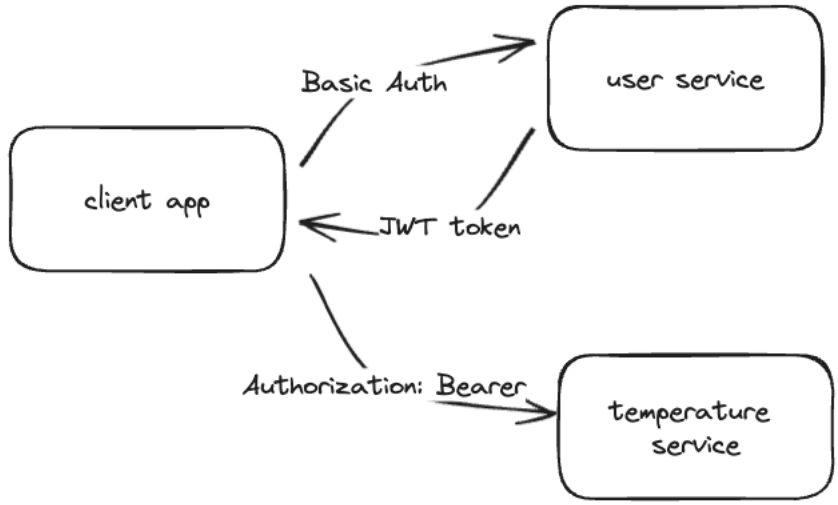
Задание 04: Аутентификация

Цель

Получение практических навыков в обеспечении безопасности систем.

Задание

Необходимо, что бы сервисы работающие с данными обрабатывали только запросы, содержащие JWT токен, аутентифицирующий пользователя.
Сервис, работы с пользователями должен реализовать endpoint «auth» который бы осуществлял basic authentication и выдавал JWT токен.



Рекомендуема последовательность выполнения работы:

- Модифицируйте REST-сервисы
- Сделайте спецификацию с OpenAPI с помощью Postman и сохраните ее в index.yml
- Протестируйте сервис
- Создайте Dockerfile для ваших сервисов
- Протестируйте его работу в Docker
- Опубликуйте на github проект

Задание 05: Data Cache

Цель

Получение практических навыков в построении сервисов, требующих высокую скорость отклика при работе с данными.

Задание

Необходимо модифицировать приложение, которое было создано в предыдущем задании. Для сущности, отвечающей за хранение клиента (пользователя) необходимо настроить настроить кеширование.
Должны выполняться следующие условия:

- Кеш должен быть реализован с помощью Redis
- Записи в кеше должны иметь срок жизни
- Необходимо реализовать шаблоны «сквозное чтение» и «сквозная запись»
- Необходимо провести сравнение время отклика и максимальной пропускной способности сервиса по запросу данных клиента (пользователя) с помощью утилиты wrk с использованием кеша и без него. Результат тестирования сохранить в репозитории в файле performance.md

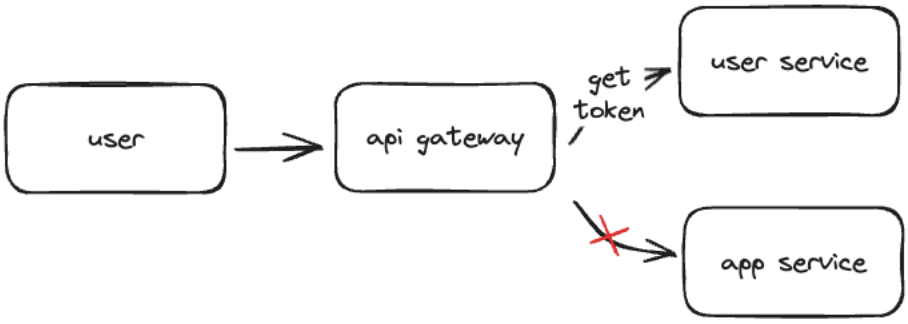
Задание 06: Circuit Breaker

Цель

Получение практических навыков в построении отказоустойчивых приложений.

Задание

Необходимо реализовать сервис API Gateway, который будет получать JWT токен в User Service и вызывать сервисы согласно варианту задания.
Для вызова сервисов, согласно варианту задания реализовать паттерн Circuit Breaker.



Форма отчетности

- Работы сдаются в виде исходного кода, опубликованного на Github.
- В репозитории должен располагаться скрипт для инициализации базы данных и заполнения тестовых значений.
- В репозитории должен располагаться файл с описанием API в формате OpenAPI
- Репозиторий должен содержать docker-compose файл для запуска приложения и dockerfile-ы для сборки его образов.
- В репозитории должно располагаться описание архитектуры проекта

Варианты

#	Название	Описание
1	Социальная сеть	Приложение должно содержать следующие данные: - Пользователь - Стена - Сообщения чата (PtP) Реализовать API: - Создание нового пользователя - Поиск пользователя по логину - Поиск пользователя по маске имя и фамилии - Добавление записи на стену - Загрузка стены пользователя - Отправка сообщения пользователю - Получение списка сообщения для пользователя
2	Магазин	Приложение должно содержать следующие данные: - Пользователь - Товар - Корзина Реализовать API: - Создание нового пользователя - Поиск пользователя по логину - Поиск пользователя по маске имя и фамилии - Создание товара - Получение списка товаров - Добавление товара в корзину - Получение корзины для пользователя
3	Сайт конференции	Приложение должно содержать следующие данные: - Пользователь - Доклад - Конференция Реализовать API: - Создание нового пользователя - Поиск пользователя по логину - Поиск пользователя по маске имя и фамилии - Создание доклада - Получение списка всех докладов - Добавление доклада в конференцию - Получение списка докладов в конференции
4	Сайт заказа услуг	Приложение должно содержать следующие данные: - Пользователь

		- Услуга - Заказ Реализовать API: - Создание нового пользователя - Поиск пользователя по логину - Поиск пользователя по маске имя и фамилии - Создание услуги - Получение списка услуг - Добавление услуг в заказ - Получение заказа для пользователя
5	Мессенджер	Приложение должно содержать следующие данные: - Пользователь - Групповой чат - PtP Чат Реализовать API: - Создание нового пользователя - Поиск пользователя по логину - Поиск пользователя по маске имя и фамилии - Создание группового чата - Добавление пользователя в чат - Добавление сообщения в групповой чат - Загрузка сообщений группового чата - Отправка PtP сообщения пользователю - Получение PtP списка сообщения для пользователя
6	Сервис доставки	Приложение должно содержать следующие данные: - Пользователь - Посылка - Доставка Реализовать API: - Создание нового пользователя - Поиск пользователя по логину - Поиск пользователя по маске имя и фамилии - Создание посылки - Получение посылок пользователя - Создание доставки от пользователя к пользователю - Получение информации о доставке по получателю - Получение информации о доставке по отправителю
7	Сервис поиска попутчиков	Приложение должно содержать следующие данные: - Пользователь - Маршрут - Поездка Реализовать API: - Создание нового пользователя - Поиск пользователя по логину - Поиск пользователя по маске имя и фамилии - Создание маршрута - Получение маршрутов пользователя - Создание поездки - Подключение пользователей к поездке - Получение информации о поездке