Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

**Департамент анализа данных и машинного обучения**

**Отчёт**

по дисциплине «Управление качеством программных систем»

**Лабораторная работа 6**

Выполнил:

студент учебной группы ПИ21-3 Шарипов Николай Ильгизарович

Факультет *«Информационные технологии и анализ больших данных»*

Направление подготовки: *09.03.03 «Прикладная информатика»*

Профиль:

*«ИТ-сервисы и технологии обработки данных в экономике и финансах»*

Научный руководитель:

Доцент, КТН

*Клочков Евгений Юрьевич*

  Москва 2024

Отчет по лабораторной работе №6

ЛАБОРАТОРНАЯ РАБОТА №6.

Тема лабораторной работы: автоматизация тестирования.

Цель работы: Разработка и реализация комплекса автотестов для проверки системы микробатчевой обработки данных в рамках микросервисной архитектуры ресторана, используя облачные технологии. Целью является обеспечение соответствия системы требованиям бизнеса, гарантия её надежности и эффективности.

Описание реализованных автотестов: инструменты, подходы

1. Pytest и Requests:
   * Pytest - фреймворк для написания и выполнения тестов на Python, обеспечивающий поддержку сложных тестовых сценариев.
   * Requests - библиотека для удобной отправки HTTP-запросов, необходимая для тестирования REST API микросервисов.
   * Подход: Создание интеграционных тестов для проверки взаимодействия между микросервисами и корректности работы API-эндпоинтов.
2. Mock и UnitTest:
   * Mock - библиотека для имитации объектов и поведения в Python, позволяющая изолировать тестируемые компоненты от внешних зависимостей.
   * UnitTest - стандартный фреймворк Python для модульного тестирования.
   * Подход: Разработка модульных тестов для отдельных функций и классов с использованием моков для имитации внешних сервисов и API.
3. Selenium WebDriver:
   * Selenium WebDriver - инструмент для автоматизации действий в веб-браузере, подходящий для тестирования веб-интерфейсов.
   * Подход: Программирование UI-тестов для проверки пользовательского интерфейса микросервисов, включая процесс оформления заказов в ресторане.

Примеры тестов:

* Тестирование микробатчевой обработки: Проверка точности и времени обработки пакетов данных, отправляемых в микросервис.
* Тестирование API-эндпоинтов: Валидация ответов микросервиса на разнообразные запросы, в том числе симуляция ошибочных сценариев.
* Тестирование пользовательского интерфейса: Имитация действий пользователя, проверка корректности отображения данных и функционирования элементов управления.

Эти автотесты направлены на обеспечение стабильности и производительности микросервисной архитектуры, а также на удовлетворение специфических требований бизнеса в сфере ресторанного обслуживания.

Код автотестов:

A screenshot of a computer program

Description automatically generated

### Ответ о тестировании (выполненные тест-кейсы, результат, выявленные дефекты):

**Тест-кейс A: Тестирование успешной микробатчевой обработки**

* **Цель:** Проверить успешную обработку микробатча данных.
* **Шаги:**
  1. Отправить запрос на микросервис с корректными данными микробатча.
  2. Получить и проверить ответ от микросервиса.
* **Ожидаемый результат:** Микросервис возвращает статус 200 и идентификатор обработанного микробатча.
* **Результат:** Тест прошел успешно. Микросервис вернул ожидаемый ответ.
* **Выявленные дефекты:** Не выявлено.
* **Заключение:** Микросервис корректно обрабатывает микробатчи данных.

**Тест-кейс B: Тестирование обработки микробатча с некорректными данными**

* **Цель:** Проверить реакцию микросервиса на некорректные данные микробатча.
* **Шаги:**
  1. Отправить запрос на микросервис с некорректными данными микробатча.
  2. Получить и проверить ответ от микросервиса.
* **Ожидаемый результат:** Микросервис возвращает статус ошибки 400 и сообщение об ошибке.
* **Результат:** Тест прошел успешно. Микросервис вернул ожидаемый ответ.
* **Выявленные дефекты:** Не выявлено.
* **Заключение:** Микросервис корректно обрабатывает ситуации с некорректными данными.

**Тест-кейс C: Тестирование проверки статуса обработки микробатча**

* **Цель:** Проверить функциональность получения статуса обработки микробатча.
* **Шаги:**
  1. Отправить запрос на получение статуса обработки для существующего микробатча.
  2. Получить и проверить ответ от микросервиса.
* **Ожидаемый результат:** Микросервис возвращает статус 200 и текущий статус обработки микробатча.
* **Результат:** Тест прошел успешно. Микросервис вернул ожидаемый статус обработки.
* **Выявленные дефекты:** Не выявлено.
* **Заключение:** Микросервис предоставляет актуальную информацию о статусе обработки микробатчей.

### Выводы по работе

В ходе выполнения лабораторной работы были разработаны автотесты для проверки микросервисной архитектуры, предназначенной для микробатчевой обработки данных в бизнес-среде ресторана. Использование современных инструментов, таких как pytest и requests, позволило эффективно тестировать функциональность и надежность системы.

Позитивные тест-кейсы подтвердили работоспособность ключевых компонентов архитектуры, включая успешную обработку микробатчей и корректное отображение статусов обработки. Однако в процессе тестирования были выявлены и некоторые дефекты, связанные с обработкой некорректных данных, что указывает на необходимость дальнейшего улучшения системы валидации входящих данных.

Результаты тестирования выявили как сильные, так и слабые стороны разработанной системы, что предоставляет ценную информацию для её оптимизации и повышения качества обслуживания клиентов ресторана. Важно продолжить тестирование для охвата более широкого спектра сценариев использования, что позволит своевременно обнаруживать и устранять потенциальные проблемы, а также гарантировать стабильность работы приложения после внесения изменений.

Список источников:

* 1. Manage Kafka Consumer Groups // Baeldung - helps developers explore the Java ecosystem and simply be better engineers. URL: https://www.baeldung.com/kafka-manage-consumer-groups.
  2. Designing Data-Intensive Applications by Martin Kleppmann // Reliable, Scalable and Maintainable Applications. – 2017, p. 3-110.
  3. Joe Reis & Matt Housley – Fundamentals of Data Engineering // Designing Good Data Architecture. – 2022, p. 111-170.
  4. Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д. – Технология разработки программного обеспечения. – 2008, с. 178-244.
  5. Pytest Documentation: <https://docs.pytest.org/en/stable/>
  6. Requests Library Documentation: <https://requests.readthedocs.io/en/master/>
  7. Mock Library Documentation: <https://docs.python.org/3/library/unittest.mock.html>
  8. Selenium WebDriver Documentation: https://www.selenium.dev/documentation/en/webdriver/