**Лабораторные работы ПнаЯВУ**

*«22. Сервис должен принимать три параметра (значение 1, значение 2, значение 3) и вернуть максимальное из предоставленных значений»*

1. Intro  
   1. Создать и запустить локально простой веб/REST сервис, используя любой открытый (например, в открытом доступе в сети интернет) пример с использованием Java stack: Spring (Spring Boot)/maven/gradle/Jersey/ Spring MVC. 2. Добавить GET эндпоинт, принимающий входные параметры в качестве queryParams в URL и возвращающий результат в виде JSON согласно варианту.
2. Error logging/handling  
   1. Добавить валидацию входных параметров с возвращением 400 ошибки, 2. Добавить обработку внутренних unchecked ошибок с возвратом 500 ошибки 3. Добавить логирование действий и ошибок 4. Написать unit test (JTest)
3. Collections intro, project structure   
   1. Добавить простейший кэш в виде in-memory Map для сервиса. Map должна содержаться в отдельном бине/классе, который должен добавляться в основной сервис с помощью dependency injection механизм Spring
4. Concurrency   
   1. Добавить сервис для подсчёта обращений к основному сервису. Счётчик должен быть реализован в виде отдельного класса, доступ к которому должен быть синхронизирован. 2. Используя jmeter/postman или любые другие средства сконфигурировать нагрузочный тест и убедиться, что счётчик обращений работает правильно при большой нагрузке.
5. Functional programming with Java 8  
   1. Преобразовать исходный сервис для работы со списком параметров для bulk операций используя Java 8 лямбда выражения. 2. Добавить POST метод для вызова bulk операции и передачи списка параметров в виде JSON
6. Functional filtering and mapping  
   1. Добавить аггрегирующий функционал (подсчёт макс, мин, средних значений) для входных параметров и результатов с использованием Java 8 map/filters функций. Расширить результат POST соответственно.
7. Data persistence  
   1. Добавить возможность сохранения всех результатов вычислений в базе данных или файле, используя стандартные persistence фреймворки Java (Spring Data/Hibernate/MyBatis)
8. Asynchronous calls  
   1. Добавить возможность асинхронного вызова сервиса используя future, возвращать статус вызова REST сервиса не дожидаясь результатов подсчётов. Результаты подсчётов должны быть представлены в БД по предопределённой ID