| ТЗ по проектной работе |  |
| --- | --- |
| **Введение**  Кратко про итоги курса, про компетенции и навыки, которые студент получил на курсе. | Тему можно выбрать самостоятельно. Главное, чтобы при реализации были применены знания по темам (и указаны на защите) **минимум от 60%** всех уроков. Тема считается примененной в проекте, если та или иная кодовая конструкция, метод решения, инструмент, подход, библиотека, фреймворк и прочее, что упоминалось в лекции или является её расширением применено в проекте (прямо или косвенно через библиотеку и это может быть обосновано на защите) – то это считается усвоенным материалом. |
| **Тема проектной работы**  Название темы проектной работы, цель проектной работы. | **Тема проектной работы**  Реализация микросервисного приложения в готового для работы в Kubernetes  **Канал фиксирования темы:** в лично кабинете сами пишите/предлагаете тему проекта – доменная область микросервисного приложения  **Срок выбора** темы доменной области: **10.02** (т.е. на консультации вы уже показываете наработки, код, вопросы, а значит и тема уже должна быть выбрана)  **Дата защиты: 27.02** |
| **Описание задачи**  Что нужно сделать в проектной работе, смысл проектной работе, применение. | **Описание задачи**  **Цель:**   * Закрепить и продемонстрировать полученные знания и навыки * Подготовить pet-project для потенциального работодателя |
| **Техническое задание**  Пошаговая инструкция, что нужно сделать с подробным описанием шагов. | **Требования | Функционал**   1. Реализовать от 4 до 6 отдельных java приложений на базе Spring boot. От 2х до 3х моделей данных (на одной модели живут 1-2 приложения) 2. \* Каждое приложение работает со своей схемой данных, в которой должны быть справочники 3. \* Для миграции использовать Liquibase или Flyway 4. Все приложения входят как подмодули в один проект и версионируются в одном репозитории. 5. \* В отдельной папке проекта размещаются helm скрипты для деплоя каждого приложения (в подпапках) 6. Должны быть тесты с применением JUnit + Mockito (TestNG) 7. \* Должны быть тесты с применением JMH 8. \* Для всех эндпоинтов должны быть запросы с применением JMeter (или другой инструмент для генерации нагрузки, для закрытия Нагрузочного Тестирования) 9. \* В приложениях должны быть кеши, для хранения справочных данных из БД. Работа с кешами подразумевает использование пакета java.util.concurrent 10. \* В приложениях должны быть метрики, чтобы можно было в мониторинге посчитать:     1. Для сервиса: rps, число обращений, latency, успешные/неуспешные завершения (+ свои метрики)     2. Для приложения: размер кешей, число обработанных ошибок (+ свои метрики) 11. \* Построить в графане дашборды для отображения метрик 12. \* Применять шаблоны отказоустойчивых сервисов:     1. При обращении к upstream сервисам делать повторы     2. При обработке запросов обеспечить защиту от перегрузки 13. \* Использовать планировщик задач для фоновых процессов 14. В репозитории должен быть Readme.md     1. Краткое описание микросервисного приложения     2. С описанием как собрать и запустить     3. Список публикуемых АПИ для потребителя     4. Use-cases по использованию этих АПИ (по этим кейсам нужно будет провести защиту приложения)   **\* - выполнение данного пункта будет учтено в общей доле использованных знаний** |
| **Форма сдачи**  В каком виде происходит сдача проекта, что нужно подготовить для защиты. | * Отдельный, публично доступный репозиторий (ваш GitHub) * Ссылка на репозиторий публикуется в Telegram группы (до защиты) * Сертификат выдается после окончания приемки проектных работ * Результат проектной работы должен соответствовать требованиям * **Защита**: в онлайне вы демонстрируете функционал приложения, запуская штучные запросы и скрипты из JMeter (или другого нагрузочного инструмента). Рассказываете какую максимальную нагрузку подавали, были ли проблемы, демонстрируете метрики и прочее. Все участники могут задавать вопросы по коду. На одну защиту 15 минут. |
| **Критерии оценки**  Основные условия, которые должны выполнить студент для получения положительной аттестации. | **Что мы будем оценивать**   * Умение развёртывания приложений в кластере через Helm * Умение организовать и провести нагрузочное тестирование * Организация логов для отслеживания прохождения запросов и времени выполнения * Умение провести профилирование приложения * Демонстрация защиты приложения от перегрузки   **Результат оценки**   * Обозначим сильные стороны * Отметим, что можно было сделать лучше |
| **Рекомендации к выполнению**  Рекомендации ПО  Список рекомендованных источников | * **Kubernetes через Docker Desktop** |