Лабораторная работа №3

“Исследование архитектурного решения”

Ход работы:

Часть 1.

Проектирование архитектуры

Для разрабатываемой системы на высоком уровне абстракций:

1. Тип приложения: Веб-приложение. Реализуется клиент-серверная архитектура.
2. Стратегия нераспределенного развертывания.
3. Обоснование выбора технологий. В нашем проекте используется следущие технологии: ReactJS – frontend; Java Spring – backend.
    a. ReactJS позволяет выразить, как приложение должно выглядеть в любой момент времени и автоматически управляет всеми обновлениями интерфейса при изменении данных приложения.
    b. Spring Framework обеспечивает комплексную модель разработки и конфигурации для любых современных приложений.

    Эти технологии обеспечивают высокую производительность и надежность приложения. Кроме того они имеют большое “community” и доступную документацию.

4. Показатели качества: безопасность, централизованный доступ к данным, простота обслуживания, надежность, обеспеченность технической поддержкой, тестируемость.
5. Обозначить пути реализации сквозной функциональности. Аутентификация и авторизация для обеспечения конфиденциальности пользовательских данных и надежной работы системы в целом.
6. Архитектура “To Be”. Приведены диаграммы компонентов “Component-view.jpg” и развертывания “Deployment-view.jpg”.

Часть 2.

1. Анализ архитектуры разрабатываемого приложения.
   В данном проекте мы придерживаемся клиент-серверной архитектуры и стратегии нераспределенного развертывания. Приложение соответствует заданным критериям качества, таким как безопасноть, надежность, простота обслуживания и т.д.
2. Обобщенное представление архитектуры полностью совпадает с приведенной в предыдущей части и приведена на рисунках “Component-view.jpg” и “Deployment-view.jpg”
3. Архитектура “As is”
   Диаграмма классов приведена на рисунке “Class-diagram.jpg”

Часть 3.

1. Сравнение архитектур “As is” и “ To be”.
   На разработку архитектуры данного приложения было выделено много времени. Поэтому на данный момент мы имеем практически полное совпадение архитетуры “To be” и “As is”.
2. Пути улучшения архитектуры. Лучшее продумывание взаимосвязей React-компонентов в результате чего рефакторинг и добавление новых возможностей приложения должны стать проще. Также мы придерживаемся и будем придерживаться единого стиля кода и наименования. В дальнейшем мы будем стараться принципов разделения функций (high cohesion & low coupling).