федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский УНИВЕРСИТЕТ информационных технологий, механики и оптики

**Факультет программной инженерии и компьютерной техники**

**Кафедра вычислительной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**ПО ДИСЦИПЛИНЕ «ПРОЕКТИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»**

Выполнил: Ванцев Александр Олегович

Группа: P3400

Преподаватель: Пенской Александр Владимирович

Санкт-Петербург

2019

1. Краткое описание проекта

Исследуемая вычислительная система (ВС) предназначается для работы с фактическими (мастер-) и версионными (оценочными, прогнозными) данными по известным логистическим объектам (ЛО) предприятий нефтеперерабатывающей промышленной области. Мастер-данные представляют собой результаты функционирования ЛО, т.е. на каком ЛО когда сколько каких нефтепродуктов (НП) было принято/отдано (сгружено/отгружено). Версионные данные рассчитываются по математической модели и прогнозируют аналогичные операции (получение, отгрузка) по НП в будущем. Пользователю предоставляется веб-интерфейс для работы с системой. Данные отображаются в этом веб-интерфейсе, но могут быть представлены и в виде Excel-отчета.

1. Краткое описание жизненного цикла проекта
   1. Предложение потенциальному Заказчику финансировать разработку ВС, которая автоматизирует часть его работы.
   2. Совместные с Заказчиком разработка списка функциональных требований к проекту, составление ТЗ, согласование.
      1. Разбивка выполнение проекта на этапы, и для каждого из них:
      2. Разработка программного кода.
      3. Внутреннее тестирование на разработческом контуре. Пока тестирование неуспешно, возврат к предыдущему пункту.
      4. Публикация решения на тестовом конутре.
      5. Презентация Заказчику. При возникновении замечаний возврат к пункту a.
      6. Публикация решения на продуктивном контуре.
   3. Поддержка существующего решения, включающая в себя:
      1. Исправление не выявленных на предыдущих этапах ошибок,
      2. Добавление нового необходимого функционала, реализуемого аналогично пунктам 2-3.
   4. Формулирование требований к версии 2.0 данного проекта и ее разработка. Новая версия предполагает настолько существенные отличия от настоящей, что не может считаться просто доработками в рамках текущего проекта. Она [новая версия] будет разрабатываться по описанному формальному сценарию, и только достижение ей п.4. позволит сказать, что жизненный цикл исследуемого проекта завершен.
2. Анализ архитектурных проблем
   1. Разработка GUI подразумевает решение о необходимости разделения веб-приложения на **back- и front-end**, а также последующий выбор **фреймворка**. С одной стороны, монолитное решение позволит иметь разработчиков, ответственных за б*о*льшую часть проекта, с другой стороны – снизит скорость разработки и увеличит взаимосвязанность компонентов ВС.
      1. Решение: действительное разделение на back- и front-end. Первый реализуется на ASP.NET WebAPI, второй – на Angular 2.0.
   2. Для отображения мастер-данных нужно получать сырые данные из некоторых источников, обработать их и сохранить результат. Значит, требуется **разработка БД**, что в свою очередь ведет к выбору ее структуры: **SQL / NoSQL**. Непосредственное получение данных из сторонних источников со стороны back-end вынужденно исключается.
      1. Решение: MS SQL – реляционная БД.
   3. Для получения версионных данных требуется расчет. Его выполнение по имеющейся математической модели очень ресурсоемко, но т.к. последняя построена квалифицированными специалистами и не предполагает существенной оптимизации, **рассчитывать на веб-сервере невозможно**. Необходимо делегировать данную задачу более мощному объекту ВС.
      1. Решение: запускать расчеты на отдельном компьютере (СК - суперкомпьютер).
   4. Из предыдущего пункта вытекает необходимость взаимодействия back-, front-end, БД и СК. Если связь между первыми двумя очевидна и осуществляется с помощью Web API, то **общение back-end, БД и СК** требует менее шаблонного решения за неимением такового (по крайней мере, широко опробованного и стандартизированного).
      1. Решение: использование очереди сообщений RabbitMQ. БД создает сообщения, back-end их обрабатывает и ставит данные на расчет в СК, а затем осуществляет обратное взаимодействие.
   5. Последовательное выполнение этапов реализации проекта требует нескольких не связанных между собой серверов, отличающихся по версии ВС, используемых БД. Возможно применение технологий DevOps для автоматизации процессов сборок и выпусков версий в пределах одного сервера и перенесения их с одного сервера на другой.
      1. Решение: используются 3 физически разных сервера: разработческий, тестовый и продуктивный. В силу относительно малой сложности проекта, DevOps на данном этапе не используется.
3. Выводы

Формулирование основных задач проекта и краткое его описание полезно для понимания области его применения, но уже на данном этапе возникают вопросы, связанные с построением ВС, ее архитектурой. Последующее описание жизненного цикла ВС позволяет взглянуть на ВС с другой стороны и обозначить новые архитектурные проблемы. Их анализ представлен в п.3 настоящего отчета, в котором также приведены возможные решения.

1. Переформулированные проблемы:
   1. Разделенное на back- и front-end приложение должно иметь возможность быстрой и простой разработки и добавления новых компонент (страниц) без влияния на существующий код.
   2. Должна обеспечиваться безопасность сайта: ограничение полномочий пользователя в зависимости от его должности.
   3. Должна быть возможность представления данных в excel-отчете, так же, как и загрузка данных в базу из excel-отчета.
   4. Расчет версионных данных должен производиться на отдельном суперкомпьютере (СК), более мощном, чем сервер, а так как запросов на расчет может быть много, необходимо распределять задачи на расчет между несколькими СК и осуществлять обратное взаимодействие.
   5. Должен быть реализован одностраничный пользовательский интерфейс, предоставляющий CRUD-функционал, возможность загружать в систему и скачивать excel-отчеты, ставить на расчет версии, получать уведомления о результатах загрузок и расчетов.
2. Анализ жизненного цикла с позиции ISO 15288:
   1. Процессы ЖЦ
      1. Процессы соглашения

|  |  |
| --- | --- |
| Приобретение | Презентация заказчику – сотруднику группы компаний – проекта, подписание договора внутри ГК. |
| Поставка | Перевод приложения на продуктивный контур – сервера, к которым предоставляется публичный доступ внутри корпоративной сети. |

* + 1. Процессы предприятия не анализируются в данной работе, так как в меньшей степени относятся к области IT.
    2. Процессы проекта

|  |  |
| --- | --- |
| Планирование проекта | Разработана начальная архитектура проекта, определены роли основных участников. |
| Оценка проекта | Еженедельные статусы – обсуждение рабочих вопросов для определения стадии развития, трудностей реализации, (пере-) распределения задач |
| Контроль проекта | Периодические показы системы заказчику |
| Принятие решений | Принимаются руководителем проекта вместе с начальниками департамента и подразделения. |
| Управление рисками | Проект не предполагает рисков, требующих специальных стратегий решений. |
| Управление конфигурацией | Определение групп пользователей, имеющих доступ к системе, и разрешенных действий для каждой. |
| Управление информацией | Написание спецификаций, документаций, разработка журнала событий (логирование данных). |

* + 1. Технические процессы

|  |  |
| --- | --- |
| Определение требований правообладателей | Определены системные и функциональные требования, основа для валидации, заключено соглашение. |
| Анализ требований | Требования заказчика формализованы и переведены на технический язык, определена основа для верификации. |
| Проектирование архитектуры | Спроектирована архитектура: описана система как отдельные функциональные элементы и способ их взаимодействия |
| Реализация | Разработка каждого системного элемента ответственным специалистом или их группой |
| Комплексирование | Построение целостной системы из отдельных элементов. Фактически этот шаг выполняется параллельно с предыдущим. |
| Верификация | Осуществляется параллельно с реализацией и комплексированием: разработка и интеграция компонентов системы сразу же тестируется для оперативного исправления несоответствий требованиям. |
| Передача | Периодический вывод системы в тестовую среду для отчета перед заказчиком и проведения им проверок, реже – на продуктив – для постепенного предоставления функционала. |
| Валидация | Происходит в процессе передачи. Заказчик подтверждает соответствие системы заявленным требованиям либо формулирует замечания. |
| Функционирование | Перевод системы в продуктивную среду. Формулирование задач на разработку новых компонент системы и исправление существующих. |
| Обслуживание | Поддержка работоспособности системы. Выполнение задач из предыдущего пункта, устранение неполадок. |
| Изъятие и списание | Система функционирует по меньшей мере до момента разработки версии 2.0, после чего часть элементов системы интегрируется в новую версию, другая – уничтожается. |

* 1. Стадии жизненного цикла: замысел, разработка, производство, применение, поддержка, списание, – соответствуют техническим процессам ЖЦ.