УТВЕРЖДАЮ

Директор Департамента
управления ИТ-проектами и интеграцией
АО «Концерн Росэнергоатом»
_______ О.Е. Шальнов
«_____»_____ 2021 г.

УТВЕРЖДАЮ

Директор по разработке
программного обеспечения
ЗАО «КРОК инкорпорейтед»
_____ И.А. Омехин
«_____»_____ 2021 г.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИЕЙ НА НОВОЙ ПЛАТФОРМЕ

АО «КОНЦЕРН РОСЭНЕРГОАТОМ»

АСУТД-2

РЕГЛАМЕНТ ДОРАБОТОК СИСТЕМЫ

17404049.63.11.11.000.003.ПН

Листов __

СОГЛАСОВАНО

Заместитель директора департамента -
начальник отдела развития ИТ
Департамента управления ИТ-проектами и интеграцией
АО «Концерн Росэнергоатом»
_____ А.П. Прокофьев
«____»_____ 2021 г.

Менеджер проекта
ЗАО «КРОК инкорпорейтед»
_____ М.В. Зорина
«____»_____ 2021 г.

ЛИСТ СОГЛАСОВАНИЯ

Фамилия, имя, отчество	**Должность**	**Подпись**	**Дата**

Аннотация

Настоящий Регламент описывает порядок выполнения работ по выполнению доработок системы в рамках проекта Автоматизированная система управления технической документацией на новой платформе (далее – АСУТД-2, Система) АО «Концерн Росэнергоатом».

Список версий

Дата	Версия	Описание изменения	Кем внесены изменения
05.03.2021	1.0	Версия документа, направленная на согласование	А.А. Дубонос
23.03.2021	2.0	Версия документа с учетом замечаний к версии 1 в соответствии с реестром замечаний	А.А. Дубонос

Содержание

1. Описание основных процессов

1.1. Общие положения

Необходимость внесения изменений в Систему определяется договором между Исполнителем и Заказчиком, или другим правоустанавливающим документом.

1.2. Роли участников процесса

В процессе разработки участвуют следующие функциональные роли:

1. Технический руководитель проекта, который отвечает за применяемые технические решения, качество доработок Системы, их соответствие проектным решениям (далее ПР), а также за сроки выполнения доработок Системы.
2. Инженер по разработке, который отвечает за разработку функциональности Системы.
3. Инженер по тестированию, который отвечает за проверку качества разработанной функциональности.
4. Системный инженер, который отвечает за подготовку и сопровождение вычислительной инфраструктуры, необходимой для доработки Системы.
5. Системный аналитик, который отвечает за сбор и анализ требований, а также за их фиксацию в ПР.
6. Специалист по внедрению, который отвечает за обучение, подготовку эксплуатационной документации и консультации в рамках опытной эксплуатации доработок Системы.
7. Технический писатель, который отвечает за соответствие подготавливаемых документов принятым стандартам.

1.3. Архитектура процесса

Процесс по выполнению доработок Системы должен соответствовать договору, в рамках которого выполняется доработка Системы.

1.4. Используемые инструменты

Для выполнения доработок системы необходимы следующие инструменты:

1. Java версии 8 (OpenJDK версии 8, актуальная сборка).
2. Maven версии 3.5.0 или новее.

3. Node.js версии 12.8.4 или новее.
4. Docker Desktop версии 3.2.1 или новее.
5. Среда разработки (далее IDE) для языка Java.
6. Camunda Modeler версии 4.4.0 или новее.
7. Система контроля версий (Git).
8. Система учета задач (Jira).
9. Система непрерывной интеграции и доставки кода (CI/CD).
10. Система управления тестовыми сценариями (TestRail).

1.5. Использование системы контроля версий

При выполнении доработок Системы используется система контроля версий исходного кода согласно следующим правилам.

1. Для каждой задачи в системе учета задач создается отдельная ветка в системе контроля версий, в которую разработчик помещает разработанный в рамках нее исходный код.
    а. название ветки должно совпадать с номером задачи;
        - в случае, если по задаче требуется создать вторую ветку, ветки создаются по шаблону "<номер задачи>-<порядковый номер дополнительной ветки>".
    б. по решению технического руководителя проекта допускается разработка нескольких связанных задач в рамках одной ветки;
    в. инженеру по разработке желательно производить коммиты в ветку не реже раза в день (с целью минимизации потерь в случае возникновения проблем на ПК инженера по разработке).
2. Каждое изменение в исходных кодах должно выполняться в рамках задачи в системе учета задач.
3. Каждое изменение должно сопровождаться комментарием.
    а. в первой строке комментария должен быть указан номер задачи и его заголовок через пробел;
    б. второй и последующей строкой кратко описывается суть изменений.
4. После окончания разработки инженер по разработке создает запрос за слияние изменений в основную ветку.
    а. ответственного за проведение слияния назначает технический руководитель проекта;
    б. назначенный ответственный проводит выполняет кода;

- в случае обнаружения замечаний инженер по разработке проводит их исправление.

в. после устранения всех замечаний ответственный производит слияние изменений в основную ветку;

- в случае, если автоматическое слияние изменений невозможно, ответственный сообщает об этом инженеру по разработке;
- инженер по разработке проводит слияние из основной ветки в разрабатываемую, решая все конфликты и оповещает ответственного о его окончании;
- в случае внесения существенного числа изменений ответственный может принять решение о необходимости проверки кода с учетом внесенных изменений при слиянии в разрабатываемую ветку.

1.6. Требования к исходному коду

1.6.1. Оформление исходных кодов на Java

Для оформления исходных кодов на языке Java используются следующие правила.

1. Исходные коды должны быть оформлены согласно правилам по оформлению исходного кода на Java.
2. У каждого инженера по разработке в IDE должен быть настроен плагин для проверки стиля (конфигурационный файл расположен в репозитории исходных кодов).
3. У каждого инженера по разработке в IDE должен быть настроен плагин для форматирования кода (конфигурационный файл расположен в репозитории исходных кодов).
4. Исходный код проекта должен проходить проверку статическим анализатором кода (настроенным для проекта) без ошибок и уведомлений.
    a. в случае ложного срабатывания сообщение должно быть отключено с помощью соответствующей аннотации. Рядом обязательно наличие комментария с пояснениями, почему инженер по разработке счел это ложным срабатыванием.
5. Для каждого создаваемого пакета необходимо добавлять Javadoc на уровне пакетов при помощи package-info.java.
    a. у всех пакетов (за исключением пакетов, содержащих автосгенерированный код) должна быть проставлена аннотация для указания, что параметры и возвращаемые значения методов, а также значения полей класса не могут быть null, если это не объявлено отдельно.
6. В общем случае при использовании параметризованного типа необходимо указывать параметр.
7. Методы классов не должны возвращать null или принимать null в качестве аргумента, если это не определено внешним API или SPI. Следует использовать java.util.Optional.

а. в случае, если метод возвращает null, он должен быть помечен аннотацией javax.annotation.Nullable;

б. в случае, если аргумент метода может быть null, он должен быть помечен аннотацией javax.annotation.Nullable.

8. Особенности использования возможностей фреймворка Spring:

   а. доступ ко всем создаваемым конфигурационным параметрам должен осуществляться через объект, класса, помеченный аннотацией "@ConfigurationProperties", где для всех свойств должны быть указаны комментарии об их назначении. Использование "@Value" в основном коде запрещено.

9. Разрабатываемый исходный код должен быть покрыт автоматическими тестами (тестовым кодом), которые выполняются при сборке Системы. Выделяются следующие виды автоматических тестов:

   а. интеграционные, для проверки совместимости клиента с сервером Системы;

   б. модульные, для проверки логики реализованной функциональности на соответствие ПР.

1.6.2. Оформление скриптов изменения схемы и наполнения БД

Для оформления исходных кодов для наполнения БД используются следующие правила.

1. Любые изменения в схеме или развертывание начальных данных производятся с помощью скриптов на языке SQL (предпочитаемый способ) или Java.
2. Скрипты разделены на несколько типов:

   а. общие скрипты, которые будут выполнятся на всех окружениях;

   б. скрипты окружения разработчика;

   в. скрипты окружения интеграционных тестов;

   г. скрипты миграции данных из других систем.

3. Для каждого типа скриптов существуют свои правила оформления:

   а. в первой строке всех SQL скриптов должен быть комментарий с кратким описанием по его назначению;

   б. общие скрипты и скрипты окружения разработчика:

      - запрещено изменение скриптов, попавших в основную ветку в системе контроля версий. Цель этого правила обеспечить автоматического развертывание скриптов на БД при запуске приложения. Исключение может быть сделано только по согласованию с техническим руководителем проекта;

- наименование файла скрипта формируется по следующему шаблону: V<две цифры текущей мажорной версии приложения>.<две цифры текущей минорной версии приложения>.<две цифры текущего патча приложения>.<4 цифры года создания скрипта><2 цифры месяца создания скрипта><2 цифры дня создания скрипта><две цифры часа создания скрипта><две цифры минуты создания скрипта><две цифры секунд создания скрипта (точность не требуется)>__asutd_<имя, состоящее только из строчных латинских букв, отражающее суть действия, которое выполняет скрипт, в котором пробелы заменены на подчеркивание>.sql
  Суть шаблона - обеспечение последовательности развертывания скриптов.
  Пример: "V01.00.00.20191219095111__asutd_create_task_types.sql"
  В случае создания скрипта Java кодом правила остаются теми же, но точки заменяются на подчеркивания, а вместо разделения слов пробелами используется CamelCase.
  Пример:
  "V01_00_00_20200309124000__Asutd_CreateDemoDocumentForm.java";
- в общих скриптах идентификаторы должны быть случайными в формате UUID, системные имена синхронизированы со значениями констант в java-коде и пользовательские имена определяться ПР;
- в скриптах окружения разработчика допускаются осмысленные идентификаторы объектов.

в. скрипты тестового окружения:

- скрипты могут изменяться, т.к. каждый раз выполняются на чистой БД. Скрипты разбиты по функциональному признаку. Новый скрипт следует добавлять только в случае добавления новой функциональности;

- наименование файла скрипта формируется по следующему шаблону: V<две цифры текущей мажорной версии приложения>.<две цифры текущей минорной версии приложения>.<две цифры текущего патча приложения>.<4 цифры года создания скрипта><2 цифры месяца создания скрипта><2 цифры дня создания скрипта><две цифры часа создания скрипта><две цифры минуты создания скрипта><две цифры секунд создания скрипта (точность не требуется)>__asutdapptest_<имя, состоящее только из строчных латинских букв, отражающее суть действия, которое выполняет скрипт, в котором пробелы заменены на подчеркивание>.sql
  Суть шаблона - обеспечение последовательности развертывания скриптов. Пример: "V01.00.00.20191218152613__asutdapptest_create_users.sql"
  В случае создания скрипта Java кодом правила остаются теми же, но точки заменяются на подчеркивания, а вместо разделения слов пробелами используется CamelCase. Пример: "V01_00_00_20200310103400__AsutdAppTest_Create_DocumentForm.java";
- в скриптах тестового окружения не должно быть объектов со случайными идентификаторами, системными именами и пользовательскими именами.

г. в скриптах не должно быть имен объектов БД со случайными именами;

д. в скриптах названия индексов/ограничений/ключей должно формироваться по следующему шаблону: "<название таблицы>_<колонка (-и)>_<суффикс>", где суффикс один из следующих:

- pkey – первичный ключ;
- ukey – уникальный ключ;
- excl – исключительный ключ;
- idx – другие виды индексов;
- fkey – внешний ключ;
- check – проверочное ограничение;
- seq – последовательность.

1.7. Требования к документированию

Разрабатываемый исходный код должен быть документирован на уровне исходного кода.

В java-классах должны добавляться комментарии Javadoc (для классов, конструкторов, методов и полей). Проверка соответствия к комментированию исходного кода выполняется автоматически средствами плагина для среды разработки и средствами CI/CD.

В остальных текстовых файлах исходного кода должны использоваться блочные или строчные комментарии.

1.8. Порядок взаимодействия участников процесса

Все участники процесса доработки Системы взаимодействуют в рамках регулярных ежедневных совещаний по статусу разработки.

1.9. Результаты процесса

Результатом процесса доработки Системы являются:

1. Версия Системы, которая соответствует ПР на доработку Системы.
2. Комплект документации в соответствие с требованиями договора на доработку Системы.

По разработанной версии Системы выпускается релиз Системы.

1.10. Обеспечение качества

Для обеспечения надлежащего качества выполнения доработок Системы, а также для сохранения качества функциональности Системы в целом выполняется разработка тестовых сценариев, которые покрывают все разделы ПР на Систему.

Согласно разработанным тестовым сценариям выполняются следующие виды тестирования:

1. Функциональное.
2. Регрессионное.
3. Нагрузочное.
4. Кросс-браузерное.
5. Автоматическое.

Ошибки, выявленные в ходе выполнения перечисленных мероприятий, должны быть устранены, а исправленная функциональность протестирована повторно.

2. Перечень условных обозначений, сокращений и терминов

Термин, сокращение	Определение
АО	Акционерное общество
АСУТД	Автоматизированная система управления технической документацией
БД	База данных
IDE	Интегрированная среда разработки (Integrated Development Environment)
Javadoc	Стандарт для документирования классов на языке программирования Java
ПР	Проектное решение
ПК	Персональный компьютер
CI/CD	Непрерывная интеграция и доставка кода (Continuous Integration/Continuous Delivery)