Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г.И. Носова»

кафедра вычислительной техники и программирования

Практическая работа

по дисциплине «Проектная деятельность»

название разработки: «Модуль системы CSIA для проверки целостности описания сборки продукта»

Выполнил: Варламов М.Н. студент 4 курса, группа АВб-19-1

Проверил: Белявский А.Б., доцент кафедры ВТ и П, к.т.н.

Магнитогорск, 2022

Содержание

Цель проекта

В современной разработке ПО присутствует большое количество технологий и систем, которые являются вспомогательными для данного процесса. В зависимости от типа и сложности продукта используются различные разработанные продукты, но среди всего многообразия различных систем выделяются те, которые необходимы для написания практически любого программного обеспечения. К таким системам можно отнести:

- Система контроля версий (Git, SVN)
- Система управления проектом (Jira, YouTrack, Trello)

Системы контроля версий необходимы для ведения работ над одним проектом несколькими разработчиками одновременно. Они позволяют вносить изменения в проект с разных устройств, хранят различные версии продуктов с возможностью “отката”, фиксируют изменения. Также немаловажной функцией данных систем является ветвление и слияние различных модулей или частей программного кода, что очень удобно для тестирования различных новых, но не безопасных функций продукта. Такой подход упрощает ведение разработки с технической точки зрения, так как в случае ошибок можно легко откатить изменения, всегда можно посмотреть какой разработчик и когда изменил ту или иную часть программы, а также надежно хранить исходный код и не бояться возможного удаления, ведь всегда можно восстановить исходный код продукта на любую стадию разработки.

Системы управления проектом служат для ведения разработки ПО с точки зрения менеджера. Они позволяют отслеживать задачи, которые выполняют разработчики в данный момент, проставлять затраченное время и указывать пояснения к выполняемой задаче. Однако данными системами пользуются не только менеджеры продуктов, но и сами разработчики, а также клиенты, либо рядовые пользователи разрабатываемого продукта. Все эти люди могут создавать задачи различных типов, будь то ошибка в продукте, либо новый функционал. На основе данных, которыми заполняются системы управления проектом,

менеджеры составляют аналитические планы, рассчитывают приблизительное время, затраченное на разработку продукта и получают множество другой информации, полезной при составлении отчета по разрабатываемому проекту.

Из приведенной выше информации можно понять, что данные системы имеют очень близкое отношение друг к другу. Так например, при загрузке изменений в систему контроля версии можно указать номер задачи, в рамках которой велись изменения. Также можно указывать в самой задаче ее статус после внесения изменений в продукт.

На самом деле можно разработать множество различных правил похожих на те, что приведены выше, и которые будут упрощать ведение разработки программного продукта. Отсюда следует, что данные системы должны иметь интерфейсы для взаимодействия друг с другом, но так происходит не всегда. Это может происходить по многим причинам:

- Отсутствие интерфейса взаимодействия у одной из систем
- Системы могут быть устаревшими, что затрудняет их взаимодействие
- Системы могут быть слишком новыми, и данные интерфейсы могут быть еще не реализованы
- Системы могут быть закрытыми

Это лишь небольшая часть проблем, по которым взаимодействие данных систем затруднительно. По одной из этих причин внутри компании “CompassPlus”, существует продукт, разработанный на платформе “RadixWare”, под названием “CSIA”, который решает несколько задач, необходимых для упрощения администрирования разрабатываемого ПО. Одной из задач и является модуль, который позволяет “связать” системы контроля версий и управления проектом.

Реализуемый проект представляет собой модуль для “CSIA”, который служит для сбора статистических данных с системы управления проектом, их обработки и предоставлению в удобном для пользователя формате. Помимо этого данный

проект может выполнять некоторые проверочные функции, связанные с системой контроля версий.

Техническое задание

Разработать объекты в программном коде и сущности в базе данных, которые должны отражать предоставляемые данные систем контроля версий и управления проектом, а также модули, позволяющие взаимодействовать с данными системами.

Реализовать следующий список проверок полученных данных:

- Комментарий к коммиту без задачи или хоть какого–нибудь описания
- Коммит есть, а задача имеет следующие проблемы:
    - Нет версии, совпадающей с проверяемой
    - Имеет статус не Resolved
    - Нет worklog
    - Имеет тип Question
    - Отсутствует test report
    - Не заполнены поля описания бага, если тип задачи “Bug”
    - Не выставлена компонента
    - Флаг SDP имеет значение Yes
- Задача имеет версию (мажорную совпадающую с проверяемой) и закрыта (статус “Resolved”) после предыдущей сборки (на проверяемой ветке Subversion), но нет коммита на SVN

Разработать пользовательский интерфейс и настроить работу с проектом “TXPG”. Добавить возможность кастомизации проверок с помощью механизма пользовательских функций. Формировать отчет в виде HTML файла. Производить рассылку файла по электронной почте.

Разработка должна вестись на платформе RadixWare, языке программирования Java и должна представлять собой модуль программного продукта CSIA. При необходимости, дополнение базы данных должно происходить с помощью той же платформы.

Описание предметной области

Система контроля версий “Subversion”

Как было описано выше, основная проблема, которую решает реализуемый модуль – “связать” данные системы контроля версий и управления проектом. Под данным термином подразумевается выгрузка данных их обеих систем и их последующая обработка.

В качестве системы контроля версии используется “Subversion”, далее “SVN”. Данная система является клиент–серверной, что дает возможность не привлекать разработчика к процессу отправки данных, а получать все данные напрямую с сервера. Для дальнейшего понимания описания стоит немного углубиться в терминологию.

Репозиторий – является сердцем любой системы контроля версий. Это центральное место, где разработчики хранят всю свою работу. Репозиторий хранит не только файлы, но и историю. Доступ к репозиторию осуществляется через сеть, выступая в роли сервера и инструмента контроля версий, выступающего в роли клиента. Клиенты могут подключаться к хранилищу, а затем они могут сохранять или извлекать свои изменения в/из хранилища. Сохраняя изменения, клиент делает эти изменения доступными для других людей, а путем извлечения изменений клиент воспринимает изменения других людей как рабочую копию.

Фиксация изменений (далее коммит) – это процесс сохранения изменений с частного рабочего места на центральный сервер. После фиксации изменения становятся доступными для всей команды. Другие разработчики могут получить эти изменения, обновив свою рабочую копию. Фиксация – это атомарная операция. Либо весь коммит успешен, либо откатан. Пользователи никогда не видят наполовину завершенный коммит.

Ветвь (далее бранч) – это копия кода, полученная из определенной точки магистрали, которая используется для внесения существенных изменений в код при сохранении целостности кода в магистрали. Если основные изменения работают по плану, они обычно сливаются обратно в магистраль.

Во время коммита изменений есть необходимость в сообщении, которое будет кратко описывать проделанную работу. Однако в этот момент возникает первая проблема, которую призван решить разрабатываемый модуль. Сообщение к коммиту имеет свободный формат и системой контроля версий оно не проверяется. В ?нашей? компании существует определенный регламент оформления такого сообщения. Он имеет следующую структуру:

[<Тип коммита>] (<Номер задачи>) <Название модуля>. <Сообщение>

Например:

[.i] (TXPG-2164) Registry.RestApi. Добавлен новый Get метод listUsers

Таким образом, как уже упоминалось выше, необходимо разработать функцию валидации сообщения к коммиту.

Из примера можно увидеть, что в сообщении мы можем определить номер задачи, над которой велась работа в продукте. Это будет ключевым звеном для получение информации из системы управления проектом. Также, мы можем узнать и модуль, в котором велась разработка, данный параметр тоже необходимо сравнивать с тем, который указан в системе управления проектом.

Помимо сообщений коммитов необходимо учитывать различные бранчи продукта. Номер бранча не хранится в сообщении коммита. Это значение берётся из той директории, в которой мы находимся в структуре проекта, и которая является бранчем. Данная информация также нужна при проверке и может быть полезна как стартовая точка начала проверки, т.к. проверка всего репозитория является крайне дорогой процедурой.

Для получения всей вышеописанной информации была использована библиотека, написанная на языке Java – SVNKit. Это чистый инструментарий для Java он реализует все функции Subversion и предоставляет API для работы с рабочими копиями Subversion, доступа и управления репозиториями Subversion – все в Java–приложении.

Система управления проектом “Atlassian Jira”

Перейдем к системе управления проектом. В нашем случае, в роли данной системы, выступает продукт “Atlassian Jira”. Платформа для управления проектами, задачами и отслеживания ошибок. Данная платформа предназначена, в первую очередь, для разработчиков и ведения agile–проектов. Jira доступна в веб–версии и в виде десктопного приложения.

Платформа JIRA представляет собой гибкий инструмент, в котором компании могут адаптировать и изменять под свои нужды функциональный внешний вид сервиса. Для каждого отдельного проекта администратор Jira может определить тип задач с уникальными составляющими элементами, привязать набор статусов и другое. К каждому проекту можно определить права доступа для участников.

Структура JIRA состоит из трёх элементов: проект, задачи и подзадачи. Проект – основной элемент платформы, в котором хранятся задачи и информация по работе над программой. Пользователи имеют возможность создать проект с нуля или использовать готовый шаблон. Для отслеживания хода работы над проектом автоматически создается дорожная карта. Дорожная карта проекта представляет собой иерархическую структуру, позволяющую планировать рабочий процесс в разных временных перспективах, отслеживать процесс выполнения задач и систематизировать работу нескольких команд над одним проектом.

Задачи JIRA – это структурированные инструменты для управление проектом. В задачах содержится информация о необходимых действиях, фиксируется время для ее выполнения, устанавливается исполнитель, прикрепляются

дополнительные файлы. Пользователь может получать уведомления о внесенных изменениях задачи, вести журнал выполнения, создавать подзадачи и оставлять комментарии.

Информация, хранящаяся в задаче на Jira должна совпадать с той, что указана в SVN коммите.

Для получения данной информации был использован HTTP–клиент, предоставляемый платформой RadixWare. Помимо этого, была написана собственная “обёртка”, для упрощения работы и более удобного получения данных.

Чтобы понять, как именно реализовано получение данных – нужно разобраться с понятием RestAPI.

RestAPI

REST API – это способ взаимодействия сайтов и веб–приложений с сервером. Его также называют RESTful.

Термин состоит из двух аббревиатур, которые расшифровываются следующим образом. API (Application Programming Interface) – это код, который позволяет двум приложениям обмениваться данными с сервера. На русском языке его принято называть программным интерфейсом приложения. REST (Representational State Transfer) – это способ создания API с помощью протокола HTTP. На русском его называют «передачей состояния представления».

Технологию REST API применяют везде, где пользователю сайта или веб–приложения нужно предоставить данные с сервера. Например, при нажатии иконки с видео на видеохостинге REST API проводит операции и запускает ролик с сервера в браузере. В настоящее время это самый распространенный способ организации API. Он вытеснил ранее популярные способы SOAP и WSDL. В нашем случае, форматом передачи данных является JSON.

У RESTful нет единого стандарта работы: его называют «архитектурным стилем» для операций по работе с серверов. Такой подход в 2000 году в своей диссертации ввел программист и исследователь Рой Филдинг, один из создателей протокола HTTP.

Проектные решения

Основные функции модуля

К основным функциям модуля относится следующее:

- Проверки различных данных
- Формирование HTML отчёта
- Рассылка по электронной почте

Выполнение функций происходит последовательно, в том порядке, в котором они описаны. Рассылка по электронной почте настраивается отдельно, с возможностью отключения.

При проверках также присутствуют дополнительные возможности кастомизации с помощью механизма пользовательских функций. Для каждой проверки существует свой ряд настроек, которые можно менять.

Формирование отчета кастомизации не подвергается, однако планируется добавить поддержку создания собственного шаблона.

Информационные потоки

На рисунке 1 представлены информационные потоки.

Рисунок 1 - Информационные потоки модуля

Все информационные потоки обрабатываются последовательно. Порядок обработки зависит от конкретной проверки.

Схема взаимодействия основных функций системы

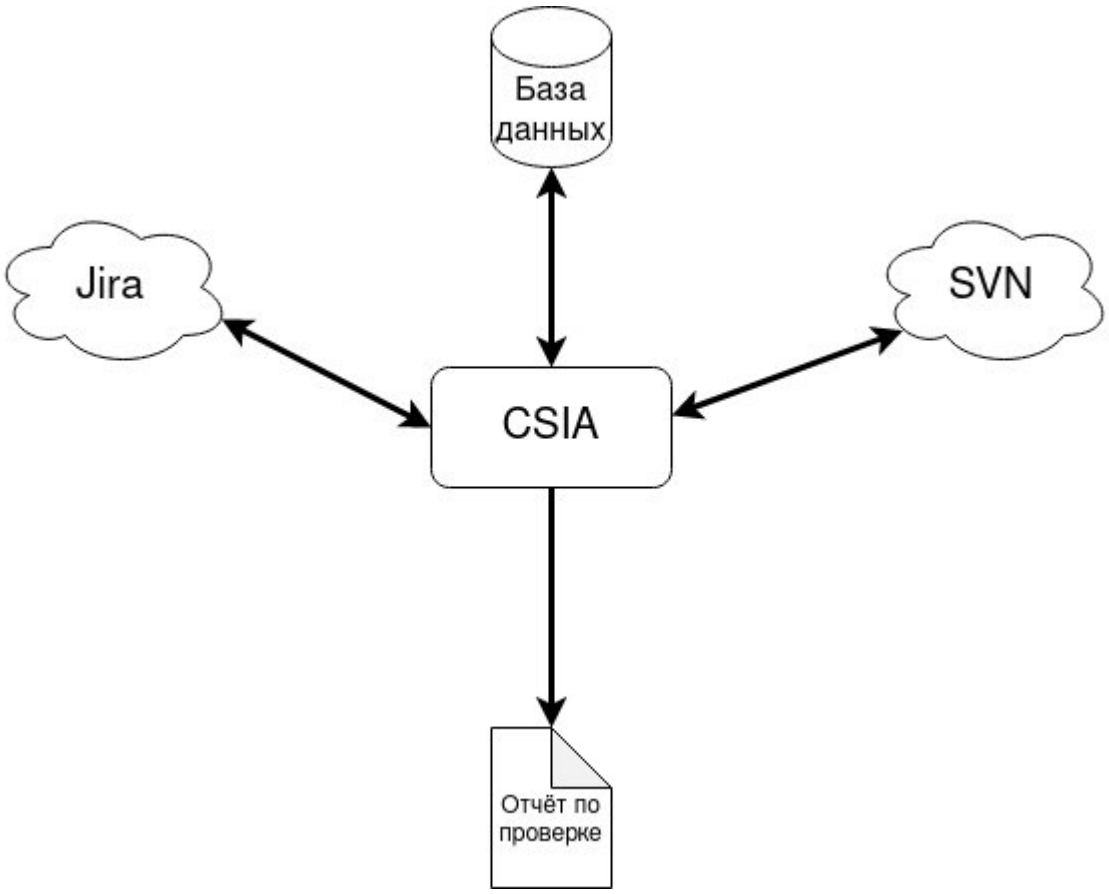


Рисунок 2 - Схема взаимодействия основных функций системы

В основе схемы (рис. 2) взаимодействия лежит разработанный модуль, который обращается к информационным потокам по мере необходимости. Последовательность взаимодействия определяется конкретным правилом и настройками, которые были выбраны пользователем.

Схема вариантов использования

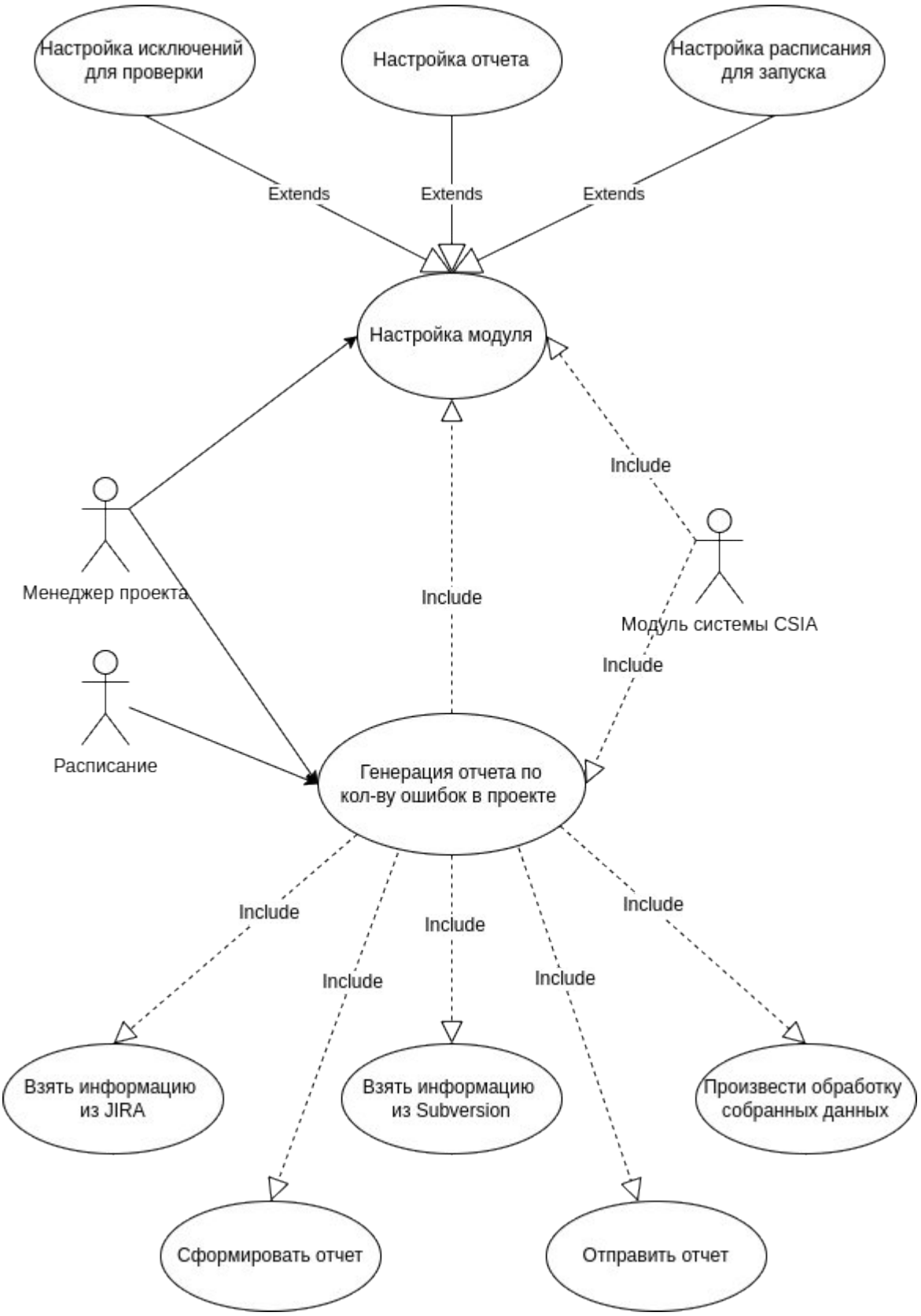


Рисунок 3 - Схема вариантов использования

На рисунке 3 изображена схема вариантов использования. На ней изображены 3 актёра:

- Менеджер проекта;
- Модуль системы CSIA;
- Расписание.

Тут, в роли менеджера проекта выступает непосредственно менеджер. Он производит первоначальную настройку модуля и получает специальный отчет, который включает в себя статистические данные по всему проекту.

Модуль системы CSIA – разрабатываемый модуль, который производит все операции по управлению данными и формированию отчетов.

Расписание – отдельный специальный модуль CSIA, который выполняет функцию автоматического запуска текущего модуля по расписанию.

Диаграмма IDEF0

На рисунках 4 и 5 представлена диаграмма IDEF0.

В качестве входных данных выступает информация о задачах и коммитах. Данная информация берется из соответствующих систем управления проектами и систем контроля версий.

Механизмом является система CSIA. Ее модули предоставляют различные способы для обработки данных, их управлением, сбором и т.д.

Правила оформления коммитов и задач являются контролем в данной диаграмме. Данные правила являются внутренними для компании.

Результатом работы модуля является отчет о нарушениях для группы разработчиков, которые ведут работу над проектом, который проверяется модулем.

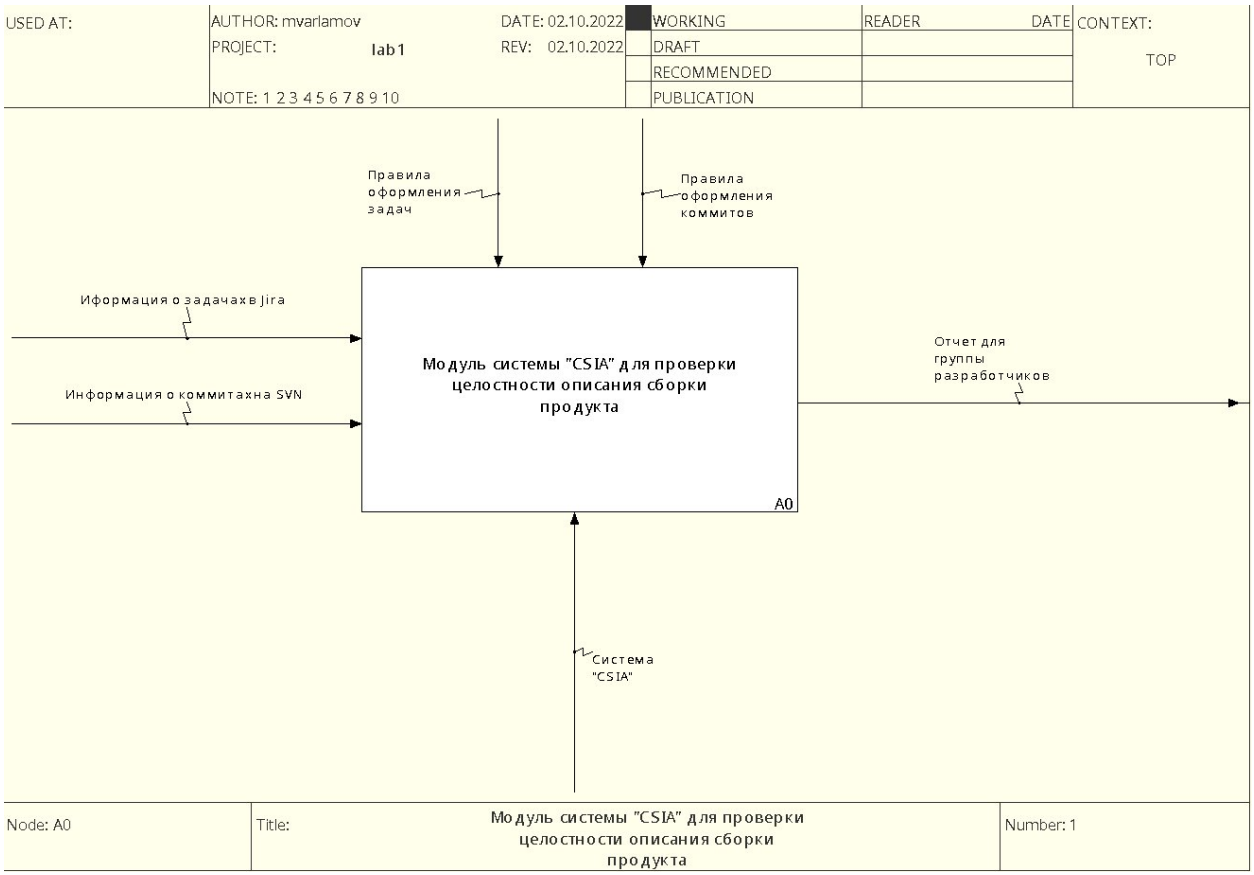


Рисунок 4 - Схема IDEF0 первый уровень

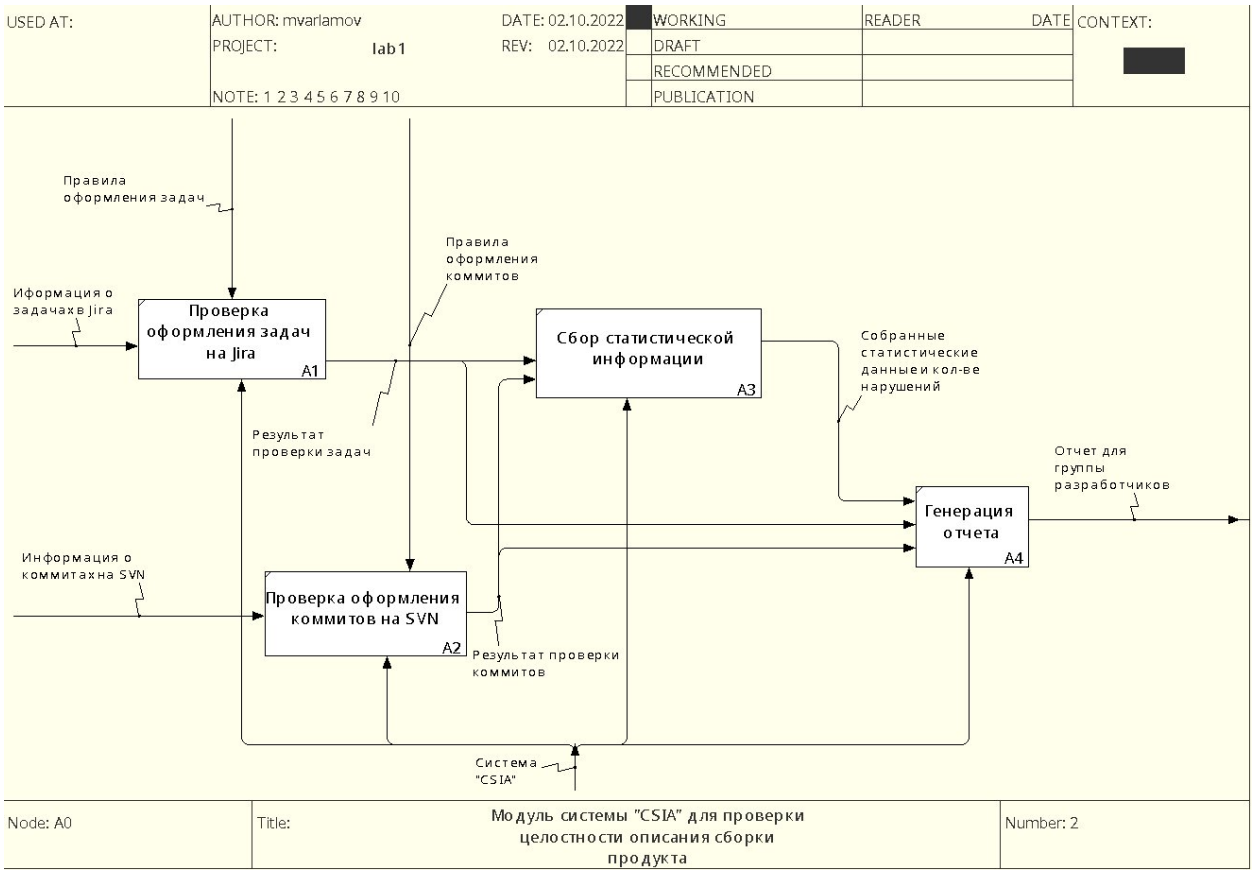


Рисунок 5 - Схема IDEF0 второй уровень

Разработка ПО

Выбор технологии

Разработка велась с помощью платформы RadixWare.

Данная платформа предназначена для разработки прикладных систем корпоративного уровня. RadixWare наиболее эффективен для разработки следующих типов приложений:

- Системы транзакций
- Системы массового обслуживания
- Высоко доступные и отказоустойчивые системы системы с трехуровневой архитектурой: БД SQL, сервер приложений (кластер), терминалы
- Системы на основе SOA (SOA, сервис-ориентированная архитектура)
- BPM-системы (Business Process Management), организация рабочих процессов

Среда программирования RadixWare Designer основана на платформе NetBeans. В соответствии с моделью продукта среда программирования генерирует байт-код Java, который будет выполняться на виртуальной машине Java в виде своевременно загружаемых частей RadixWare Server или Explorer, которые являются приложениями Java. Код SQL, PL/SQL и JavaScript генерируется для выполнения СУБД и RadixWare Web Explorer соответственно. В трехуровневой архитектуре этот код выполняется на сервере приложений и на клиенте. Рабочие места могут быть организованы одним из следующих способов: С помощью клиентского приложения RadixWare Desktop Explorer. Пользовательский интерфейс клиентского приложения использует библиотеку QtJambi. Технология Java Web Start может использоваться для автоматизированной доставки

компонента запуска RadixWare Desktop Explorer на все рабочие места по протоколу HTTP (HTTPS). Платформа RadixWare обеспечивает автоматическое обновление компонентов клиентского приложения, за исключением компонента запуска, который обновляется вручную или с помощью Java Web StartSoftware. С помощью клиентского приложения RadixWare Web Explorer. Приложение RadixWare Web Presentation Server используется для обеспечения доступа к рабочему месту по протоколу HTTPS (или HTTP), а браузер используется в качестве клиентского приложения. Для обеспечения оперативного обновления версий продукта код приложения хранится в репозитории Subversion.

Разработка модуля

Разработка модуля велась в несколько этапов. К первому этапу можно отнести разработку необходимых сущностей в базе данных. Для реализации заявленного функционала необходимо было дополнить существующую объектную модель следующими сущностями:

- WorkLog
- InactivityLog

В результате, объектная модель представлена на рисунке 5.

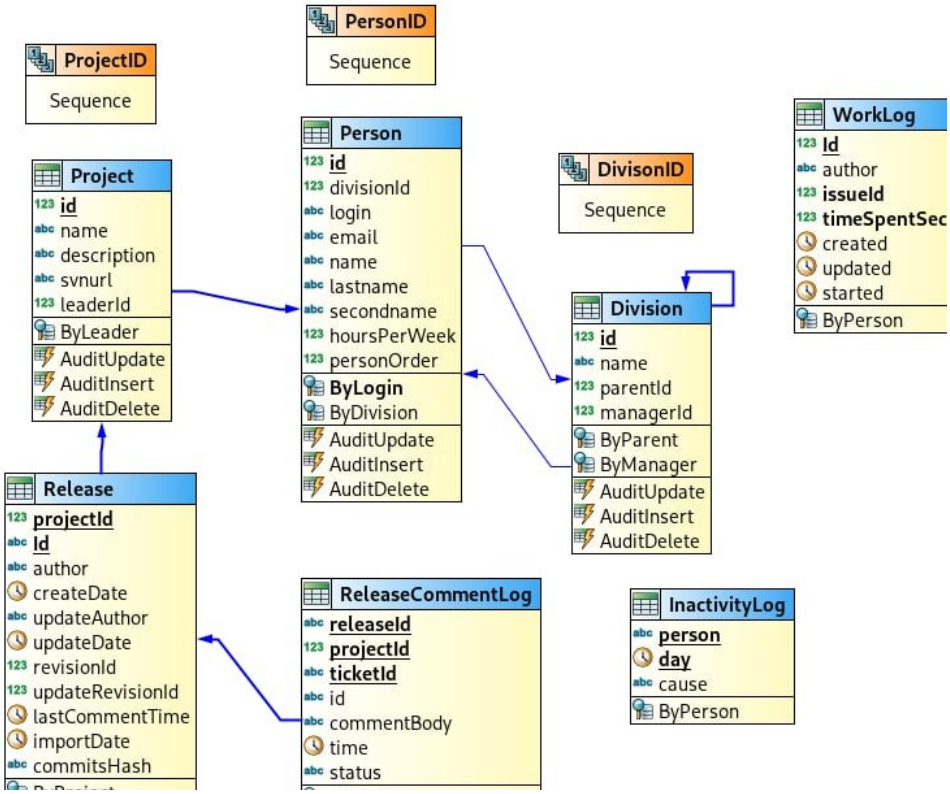


Рисунок 5 - Объектная модель продукта

Разработка модели выполнялась с помощью специального конструктора (рис. 6, 7).

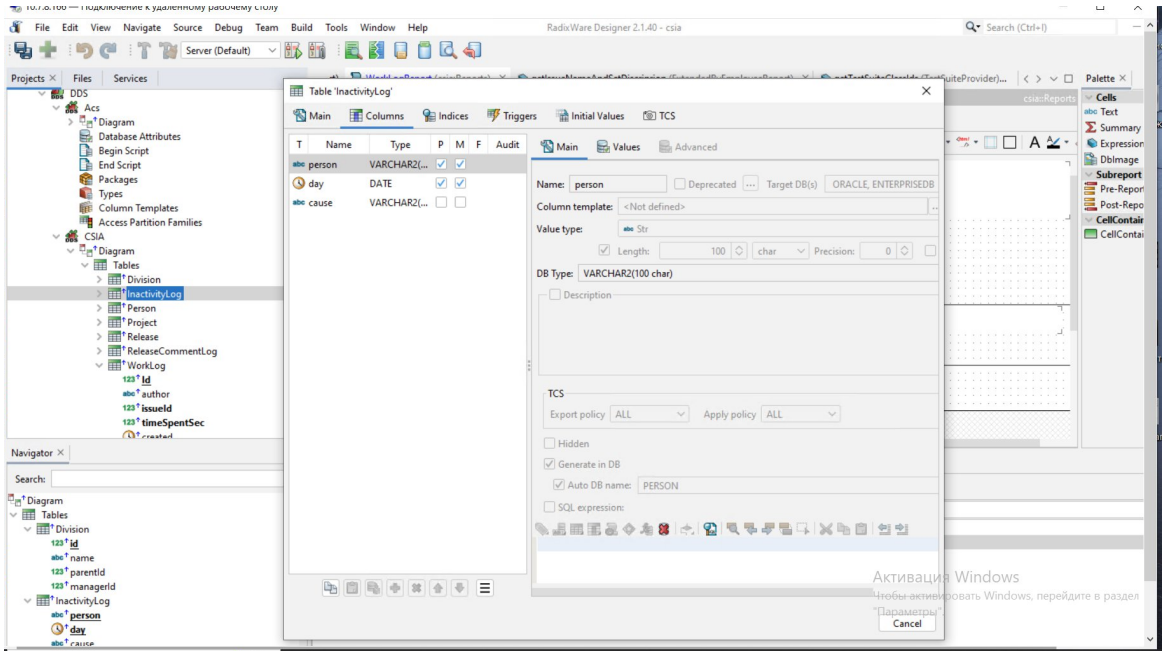


Рисунок 6 - Дизайнер базы данных

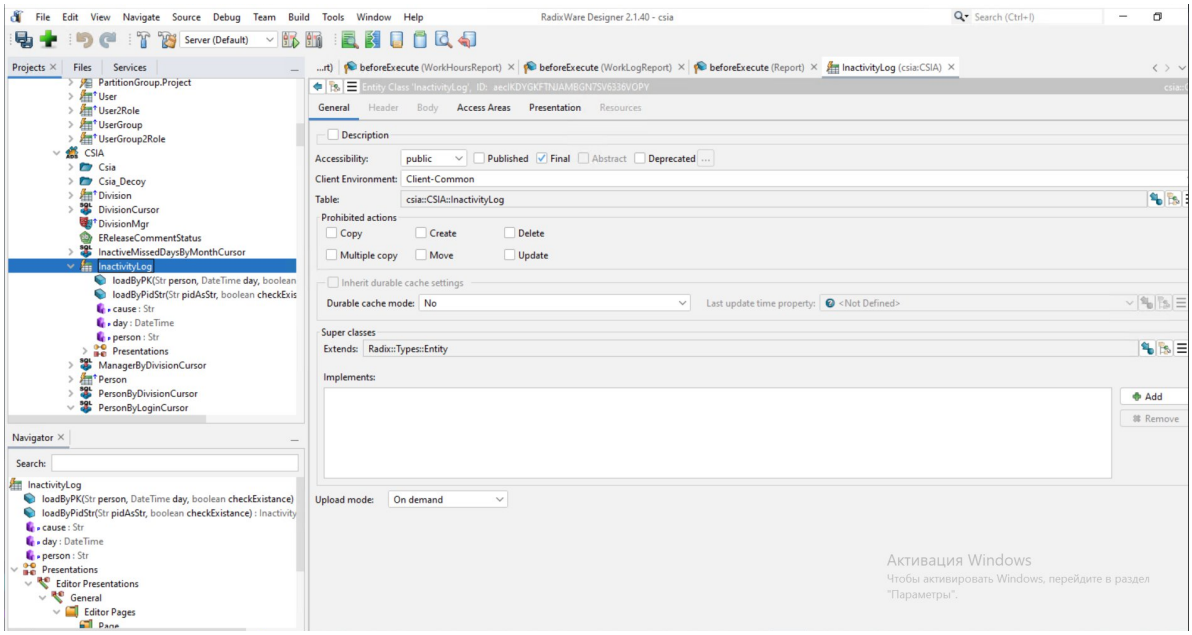


Рисунок 7 - Дизайнер форм

Далее, на основе созданных сущностей в DDS слое продукта, были созданы уже программные объекты в слое ADS. Были перенесены все необходимые поля, созданы методы для загрузки данных из БД.

Ко второму этапу можно отнести создание презентации редактора и селектора созданных сущностей. Это нужно для удобного управления сущностями из графического интерфейса продукта. Результат разработки графического интерфейса представлен на рисунке 8 и 9.

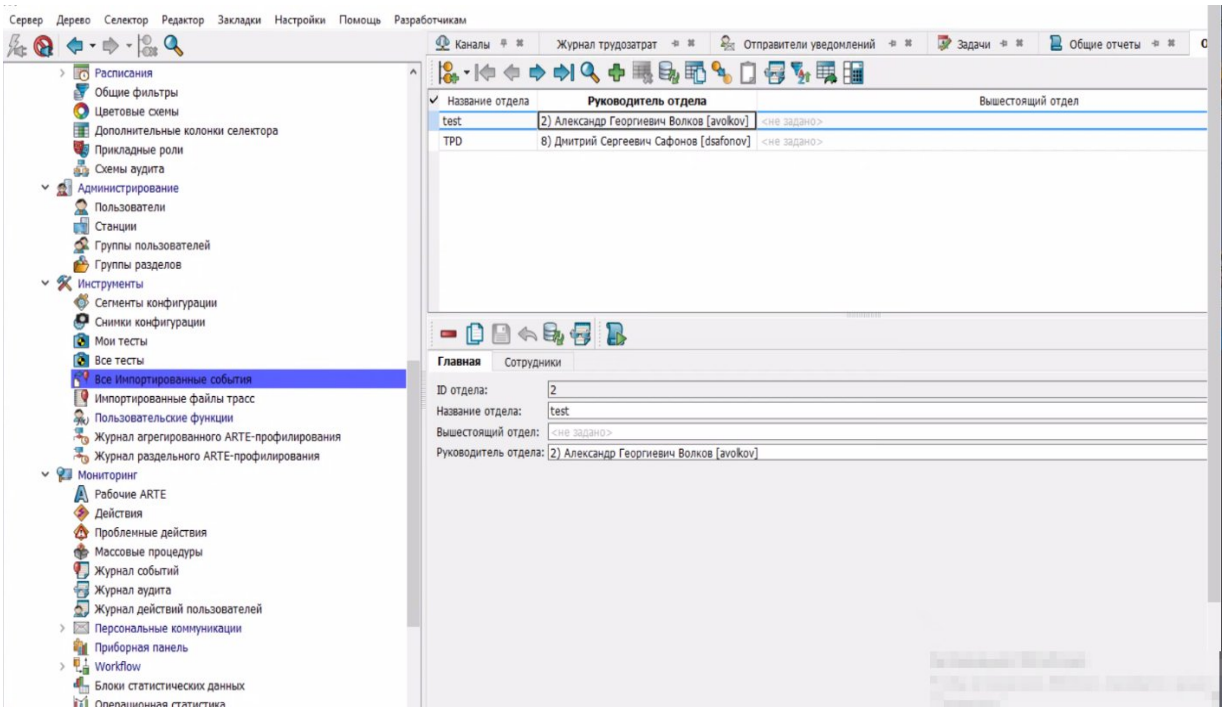


Рисунок 8 - Разработанный графический интерфейс

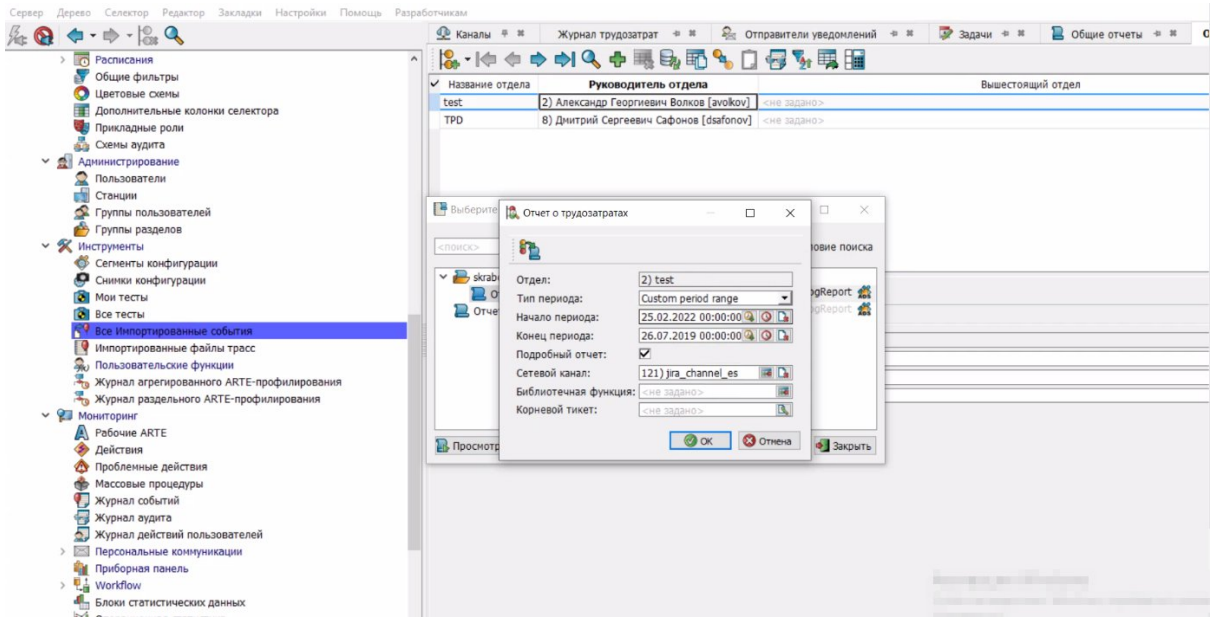


Рисунок 9 - Разработанный графический интерфейс

К третьему этапу относится разработка необходимых модулей для связи с системами контроля версий и управления проектами.

Для создания модуля связи с системой контроля версий использовалась библиотека SVNKit. SVNKit - это чистый инструментарий Java - он реализует все функции Subversion и предоставляет API для работы с рабочими копиями Subversion, доступа и управления репозиториями Subversion. SVNKit написан на Java и не требует дополнительных двоичных файлов или собственных приложений. Он переносим и не требует специального кода для ОС, что позволяет не перегружать проект различными зависимостями.

С помощью SVNKit был написал утилитарный класс, который реализует следующие функции:

- Взять информацию о коммите
- Взять информацию о всех коммитах ветки
- Взять информацию о всех ветках

Полученные данные сравниваются с теми, что были получены с системы управления проектами.

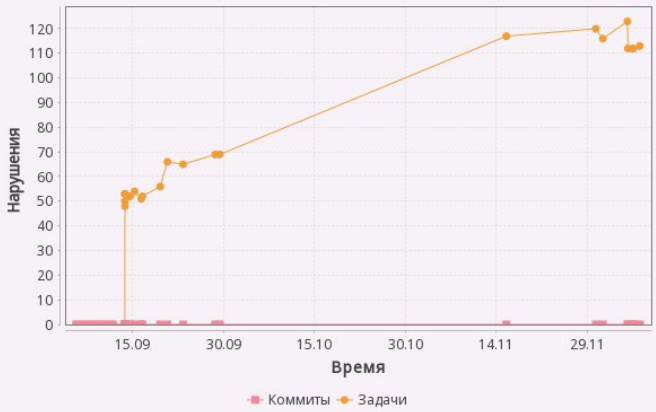
В свою очередь эти данные поступают в реализованный модуль с помощью утилитарного класса “JiraHandler”. Данный класс реализует следующие функции:

- Взять информацию о задаче
- Взять информацию о всех задачах проекта
- Взять информацию о всех проектах

В результате, все полученные данные поступают в класс обработчик, который сравнивает и сопоставляет информацию из двух источников и на их основе выдает результат в виде HTML файла. Пример такого файла изображен на рисунке 10.

Отчет по TXPG для dchernienko

Динамика нарушений



▼ Ошибки в оформлени задач

№	Issue	Violation type	Message
1	TXPG-3501	NO_COMPONENTS	Components not specified
2	TXPG-3518	NO_TEST_REPORT	Issue has a 'Resolved' status, but no test report
3		NO_BUG_DESCRIPTION	Issue type is 'Bug', but no bug description or condition or temp. decision
4	TXPG-3523	NO_TEST_REPORT	Issue has a 'Resolved' status, but no test report
5	TXPG-3526	NO_REVIEW_SUBTASK	Task is need in review, but review subtask not found
6		NO_ORIGINAL_ESTIMATE	Client is specified ( ), issue type is 'Task', but no original estimate.
7		NO_COMPONENTS	Components not specified
8	TXPG-3533	NO_REVIEW_SUBTASK	Task is need in review, but review subtask not found
9		NO_TEST_REPORT	Issue has a 'Resolved' status, but no test report
10		NO_ORIGINAL_ESTIMATE	Client is specified ( ), issue type is 'Task', but no original estimate.
11	TXPG-3539	NO_REVIEW_SUBTASK	Task is need in review, but review subtask not found
12		NO_TEST_REPORT	Issue has a 'Resolved' status, but no test report
13		NO_ORIGINAL_ESTIMATE	Client is specified (TEST), issue type is 'Task', but no original estimate.
14	TXPG-3544	NO_REVIEW_SUBTASK	Task is need in review, but review subtask not found
15		NO_BUG_DESCRIPTION	Issue type is 'Bug', but no bug description or condition or temp. decision
16	TXPG-3549	NO_COMPONENTS	Components not specified

Рисунок 10 - Пример HTML отчёта