Выпускная квалификационная работа на тему «Разработка модуля системы CSIA для мониторинга описания работ по разработке ПО в системах контроля версий и управления проектами»

Исходные данные к работе

1. ГОСТ 34.602.89 «Техническое задание на создание автоматизированной системы»
2. СМК-О-СМГТУ-36-20 «Выпускная квалификационная работа: структура, содержание, общие правила выполнения и оформления»
3. Документация к системе RadixWare компании ООО «Компас Плюс»
4. Структура системы контроля версий «Subversion»
5. Структура системы контроля версий «GitTea»
6. Структура системы управления проектами «Jira»

Перечень вопросов, подлежащих разработке в выпускной квалификационной работе:

1. Анализ требований, предъявляемых к разрабатываемой системе
2. Изучение систем контроля версий и систем управления проектами
3. Разработка проектного решения модуля на базе системы «CSIA»
4. Разработка модуля на базе системы «CSIA»

Графическая часть: подготовка электронной презентации выпускной квалификационной работы средствами ...

Слайд 1 - ...

...

**Реферат**

Тема выпускной квалификационной работы: «Разработка модуля системы CSIA для мониторинга описания работ по разработке ПО в системах контроля версий и управления проектами».

Ключевые слова: СИСТЕМА КОНТОРЯ ВЕРСИЙ, СИСТЕМА УПРАВЛЕНИЯ ПРОЕКТАМИ, ...

Пояснительная записка содержит n страниц без приложения, n иллюстраций, n информационных источников.

Объектом исследования в данной выпускной квалификационной работе является процесс разработки программного обеспечения. Предмет исследования – описание работы в системах контроля версий и управления проектами.

Цель данной работы – разработка модуля системы CSIA для мониторинга описания работ по разработке ПО в системах контроля версий и управления проектами.

В пояснительной записке изучены особенности работы систем контроля версий и управления проектами. Рассмотрены проектные решения по разработке модуля на базе системы CSIA. ...

Отзыв

Содержание

[Введение 8](#_Toc1)

[1 Особенности разработки программных продуктов в ООО «Компас плюс» 9](#_Toc2)

[1.1 Общие характеристики коммерческих продуктов компании 9](#_Toc3)

[1.1.1 Общая архитектура продуктов 9](#_Toc4)

[1.1.2 Хостовые приложения 12](#_Toc5)

[1.1.3 Front-end приложения 13](#_Toc6)

[1.2 Системы используемые при разработке ПО 14](#_Toc7)

[1.2.1 Система управления версиями Subversion 14](#_Toc8)

[1.2.2 Система контроля версий Git 16](#_Toc9)

[1.2.3 Система управления проектами Jira 18](#_Toc10)

[1.3 Нормативы описания разработок 18](#_Toc11)

[1.3.1 Формат коммитов в системах контроля версий 18](#_Toc12)

[1.3.2 Регламенты описания задач в системах управления проектами 18](#_Toc13)

[1.4 Контроль качества продукта 18](#_Toc14)

[1.4.1 Тестирование 18](#_Toc15)

[1.4.2 Документация 19](#_Toc16)

[1.4 Выводы к первой главе 19](#_Toc17)

[2 Проектные решения разработки модуля системы csia для мониторинга описания работ по разработке по в системах контроля версий и управления проектами 20](#_Toc18)

[2.1 Общие сведения о система CSIA 20](#_Toc19)

[2.1.1 Платформа RadixWare 20](#_Toc20)

[2.1.2 Модули системы CSIA 20](#_Toc21)

[2.2 Требования к разрабатываемому модулю 20](#_Toc22)

[2.3 Организация взаимодействия с системами контроля версий 20](#_Toc23)

[2.3.1 Взаимодействие с Subverion 20](#_Toc24)

[2.3.2 Взаимодействие с GitTea 20](#_Toc25)

[2.4 Организация взаимодействия с системой управления проектами Jira 20](#_Toc26)

[2.5 Мониторинг нарушений в проекте 20](#_Toc27)

[2.5.1 Структурная модель 20](#_Toc28)

[2.5.2 Алгоритм поиска нарушений 20](#_Toc29)

[2.6 Генерация отчета 20](#_Toc30)

[2.6.1 Работа со статистикой нарушений 20](#_Toc31)

[2.6.2 Алгоритм генерации отчета 20](#_Toc32)

[2.7 Выводы ко второй главе 20](#_Toc33)

[3 Результаты опытной эксплуатации модуля системы csia для мониторинга описания работ по разработке по в системах контроля версий и управления проектами 22](#_Toc34)

[3.1 Работа с разработанными сущностями 22](#_Toc35)

[3.1.1 Сущность проекта 22](#_Toc36)

[3.1.2 Сущность репозитория 22](#_Toc37)

[3.1.3 Сущности системы управления проектами 22](#_Toc38)

[3.2 Создание и настройка задачи для мониторинга 22](#_Toc39)

[3.2.1 Задача Task.CheckProjectViolations 22](#_Toc40)

[3.2.2 Настройка расписания 22](#_Toc41)

[3.3 Пример сгенерированного отчёта 22](#_Toc42)

[3.4 Выводы к третьей главе 22](#_Toc43)

[Заключение 23](#_Toc44)

[Список используемых источников 24](#_Toc45)

[Приложение А 25](#_Toc46)

# Введение

В настоящее время программное обеспечение ...

# **1 Особенности разработки программных продуктов в ООО «Компас плюс»**

## **1.1 Общие характеристики коммерческих продуктов компании**

### 1.1.1 Общая архитектура продуктов

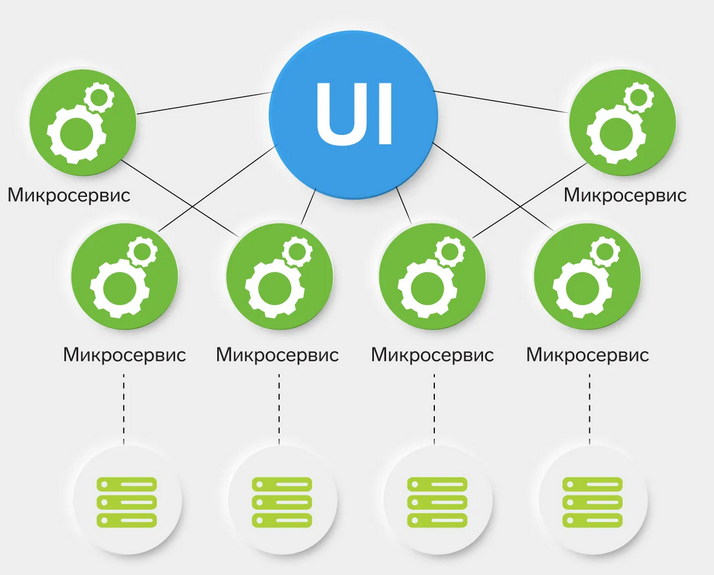
Специфика коммерческих продуктов компании предполагает большой набор критериев, которым должны соответствовать разрабатываемые информационные системы. Из основных выделяются следующие критерии:

1. Отказоустойчивость – системы, работающие в реальном времени и выполняющие услуги по денежному переводу, что является одной из основных задач продуктов компании, должны работать с минимальным количеством сбоев. Клиент должен быть уверен, что его перевод совершится, либо все связанные с переводом сущности вернутся в исходное состояние, иными словами должна обеспечиваться транзакционная целостность. Помимо этого, выход из строя одного из компонентов системы не должен приводить к неработоспособности всей системы в целом;
2. Масштабируемость – информационная система должная обладать свойствами горизонтального масштабирования. Это необходимо для распределения высокой нагрузки на систему. Сервисы системы, физически, могут находиться на различных вычислительных устройствах и использовать максимальное количество доступных им ресурсов, что увеличивает скорость выполнения запроса;
3. Эффективное выполнение задач – скорость выполнения запроса очень важна. Это улучшает пользовательский опыт взаимодействия с системой и позволяет обрабатывать большее количество запросов в единицу времени. Очевидно, что любая задача может быть выполнена на любом современном языке программирования. Однако это вовсе не означает что такая задача будет выполнена эффективно. Поэтому существует необходимость создания продукта, состоящего из ряда систем, написанных на языках программирования, которые наиболее эффективно выполняют ту или иную задачу;
4. Поддержка кодовой базы – современные программные продукты имеют весьма большую кодовую базу, которая легко может содержать несколько сотен тысяч строк кода. Поддержка таких продуктов достаточно сложная задача. Для упрощения процесса поддержки необходимо четко разделять части продукта друг от друга для того, чтобы иметь четкое представление о том, что выполняет каждый элемент разрабатываемой системы.

Одной из архитектур, хорошо подходящих под данные критерии является микросервисная архитектура.

Микросервисная архитектура – архитектура программного продукта, основанная на создании небольших, автономных, совместно работающих сервисах [1].

На рисунке 1 представлен пример микросервисной архитектуры. На нем видно, что существует система, предоставляющая графический пользовательский интерфейс (UI). Выполняя какой-либо запрос пользователя обращение не поступает только к одной системе, а в зависимости от поставленной задачи используется определенный микросервис. В зависимости от необходимости, микросервисы могут обращаться друг к другу, либо, например, иметь подключение к базе данных.



*Рисунок 1 – Пример микросервисной архитектуры*

Выход из строя какого-либо элемента в этой сети не приводит к прекращению работы всего продукта, так как каждый микросервис является независимой системой, работоспособность который не влияет на все остальные элементы. Выход из строя микросервиса может означать лишь прекращение работы определенного функционала, но никак не всего продукта.

Масштабируемость может достигается с помощью клонирования микросервиса, что приводит к распределению нагрузки и повышению скорости работы всего продукта.

Каждый микросервис может быть написан на любом языке программирования с использованием любой технологии, подходящей для конкретной задачи микросервиса. Это, как уже было сказано, также благоприятно сказывается на скорости работы продукта. Помимо этого, такой подход позволяет поделить разработку продукта между несколькими, независимыми друг от друга командами, что в свою очередь повышает скорость разработки продукта, а также его качество, так как решаемые микросервисом задачи ограничены и относятся к одной области, что позволяет сфокусироваться только на их решении.

При таком подходе, деление продукта на компоненты, имеющие ограниченную область применения, происходит само собой. Каждый микросервис проектируется и разрабатывается с ограниченной функциональностью и не может заменять другой микросервис. Это позволяет разработчику достаточно быстро разобраться в структуре работы данного сервиса и внести необходимые изменения.

Таким образом, микросервисная архитектура является наиболее подходящим архитектурным паттерном для разработки высоконагруженых программных продуктов и успешно используется на протяжении нескольких лет разработки в компании «Компас плюс».

### ***1.1.2 Хостовые приложения***

Основой любого коммерческого продукта, разрабатываемого в компании «Компас плюс», является хостовое приложение.

Хостовое приложение – программное обеспечение, выполняющее основные функции разрабатываемого продукта.

В зависимости от проекта, хостовое приложение может иметь различную функциональность, архитектуру и окружение. В данном случае, под окружением понимается ряд дополнительных систем, разработанных, как в компании, так и вне её, которые в совокупности с хостовым приложением и формируют готовый программный продукт.

В качестве основы хостового приложения используется одна из двух систем. TranzWare и RadixWare.

TranzWare – это система, использующая объектно-ориентированный подход при разработке приложения. Немного про TW.

Со временем, на смену TranzWare пришла система RadixWare. Немного про Radix.

### 1.1.3 Front-end приложения

Помимо хостового приложения, проект может состоять из ряда дополнительных систем, называемыми Front-end приложениями.

Front-end приложения – программные продукты, отвечающие за взаимодействие пользователя с системой. Данные продукты являются прослойкой между пользователем и хостовой системой. Они, с помощью пользовательского интерфейса, позволяют пользователю передавать информацию к хосту и выполнять различные операции.

В качестве основы взаимодействия с пользователем может выступать как веб-страница (в качестве примера таких сервисов могут выступать Merchat portal, HPP, Internet bank), так и мобильное приложение (Mobicash, Pocket bank).

Однако нередки случаи, когда системы, которые непосредственно предоставляют взаимодействие с пользователем являются лишь частью от общего числа Front-end приложений. К Front-end приложениям можно также отнести и вспомогательные системы для кэширования данных, системы хранения статических ресурсов, системы маршрутизации запросов с возможностью конвертации протокола взаимодействия. Все эти системы могут быть написаны с помощью различных технологий. Можно выделить основные технологии и языки программирования, используемые при разработке Front-end приложений:

1. NodeJS – асинхронная среда выполнения JavaScript кода, предназначенная для создания масштабируемых сетевых приложений;
2. React – декларативная JavaScript библиотка для создания пользовательских интерфейсов;
3. Android – программный стек на базе Linux с открытым исходным кодом, созданный для широкого спектра устройств и форм-факторов;
4. Swift – язык программирования, предназначенный для создания приложений, работающих на операционной система iOS;
5. Spring – фреймворк для языка программирования java, предназначенный для разработки веб-сервисов.

## 1.2 Системы используемые при разработке ПО

### 1.2.1 Система управления версиями Subversion

В настоящее время, разрабатываемые системы имеют огромную сложность и объемы разработки. Проектирование и реализация такой системы одним человеком просто невозможна, так как это займет очень большое количество времени и сильно снизит качество конечного продукта. Поэтому над разработкой систем большой сложности, зачастую, трудится команда из нескольких разработчиков.

В свою очередь, разработку в команде трудно представить без какой-либо системы контроля версий. Это обуславливается тем, что все разработчики должны иметь доступ к исходному коду. Должно осуществляться логирование, версионирование и хранение всех внесенных изменений. Одной из таких систем контроля версий является Subversion.

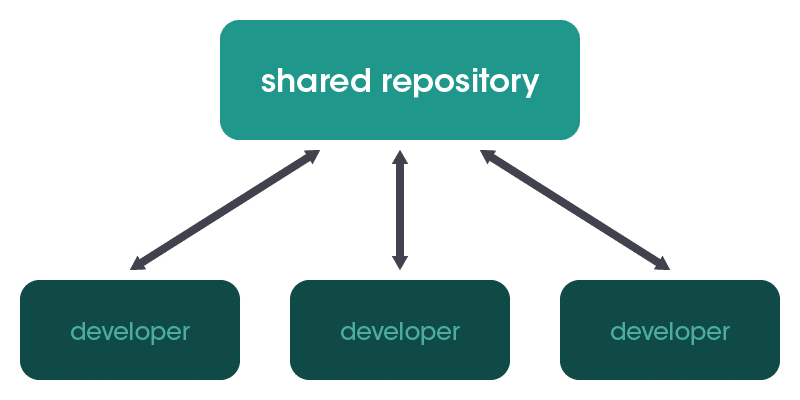
Subverison (SVN) – система контроля версий с открытым исходным кодом. Она позволяет организовать управление каталогами и файлами, а также изменениями, внесенными с течением времени. Это даёт возможность восстановить ранние версии данных и изучить историю всех изменений [2].

В 2004 году, данная система пришла как альтернатива существующей на тот момент CVS. Изначально, SVN создавался для того, чтобы исправить основные недостатки CVS, вместе с тем, сохранив высокую совместимость между этими системами.

К основным отличиям SVN от CSV можно отнести следующее:

1. Контроль изменений каталогов. CSV имеет возможность отслеживания только файлов, в то время как SVN имеет свою собственную «виртуальную» файловую систему, что позволяет этой системе отслеживать изменения каталогов;
2. Настоящая история версий. CVS позволяет контролировать только изменения в файлах. Изменений названий и копирование файлов не поддерживается. SVN же дает возможность копирования, удаления и переименование как каталога, так и файла с возможностью сохранения истории;
3. Атомарная фиксация изменений. Ряд изменений имеет возможность попасть в хранилище либо целиком, либо не попасть туда вовсе. Это предотвращает наличие ошибок вида частичного внесения изменений;
4. Выбор средства доступа к хранилищу по сети. SVN имеет абстракцию над доступом к хранилищу. Такой подход дает возможность реализации с помощью различных сетевых механизмов доступа. Можно организовать доступ как по протоколу HTTP, так и туннелировать передаваемые данные через SSH;
5. Единый способ работы с данными. Поиск различий между файлами выполняется с помощью специального бинарного алгоритма. Данный алгоритм работает одинаково как с бинарными, так и с текстовыми данными;
6. Эффективная работа с ветками и метками. Создание ветки (копии проекта с дополнительными изменениями) не должно сопровождаться полным копированием проекта и пропорциональному увеличению занимаемого на диске пространства. Subversion создает ветки с помощью механизма, похожего на механизм создания жестких ссылок в файловых системах. Благодаря этому сокращается используемое дисковое пространство и увеличивается скорость создания такой ветки [2].

В основе данной системы контроля версий лежит принцип централизованной модели (рисунок 2). Этот принцип означает наличие единственной «главной копии». Локально хранятся только те файлы, которые были изменены разработчиком. Когда разработчик вносит изменения в репозиторий это приводит к ограничению доступа для других членов команды. Этот контролируемый подход к рабочему процессу разработки снижает риск возникновения конфликтов и привлекает из-за его безопасного и централизованного характера и четкого процесса.



*Рисунок 2 – структура организации Subversion*

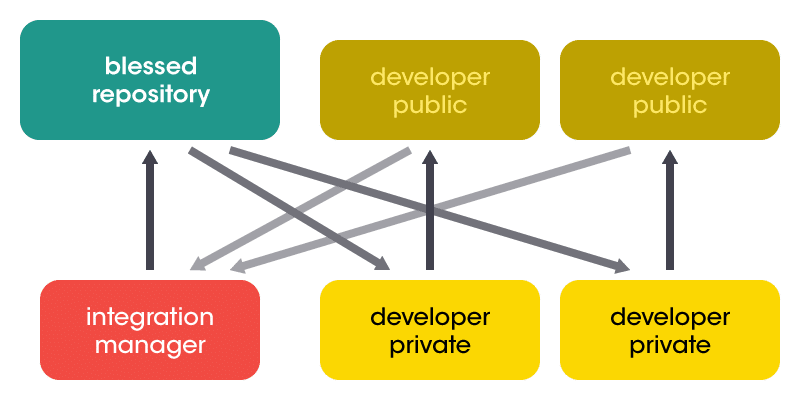
Все вышеперечисленные параметры привели к тому, что все хостовые приложения, разрабатываемые в компании «Компас плюс» используют именно эту систему контроля версий. Более того, технологические платформы, с помощью которых и ведется разработка хостовых систем имеют полную поддержку данной системы контроля версий.

### 1.2.2 Система контроля версий Git

Помимо Subversion и CSV существуют и другие системы контроля версий. Одна из таких систем является Git.

Git – распределенная система контроля версий. В таких системах каждый разработчик проекта имеет возможность создания локального репозитория, который является полной копией центрального хранилища. Благодаря такому подходу каждый разработчик может работать над проектом автономно на своем компьютере. В тот момент, когда изменения будут готовы, разработчик может отправить их в общее хранилище и объединить.

В основе системы Git, как уже было сказано лежит распределенный подход (рисунок 3). Это означает, что все разработчики имеют одинаковый доступ ко всем файлам в хранилище и каждый из них может склонировать хранимый проекта с исходным кодом и всеми метаданными. Работа с локальным репозиторием приводит к тому, что, за исключением внесения изменений, все остальные действия могут быть выполнены быстро, поскольку они затрагивают только файлы на локальном диске разработчика, а не на удаленном сервере, как это происходит в SVN. Это позволяет нескольким членам команды получать доступ к одному и тому же файлу и вносить изменения по мере необходимости, но это удобство сопряжено с риском конфликтов слияния Git. Несмотря на это, разработчики по всему миру понимают ценность распределенной модели, поскольку она упрощает совместное использование кода и облегчает удобную совместную работу с большими группами.



*Рисунок 3 – структура работы Git*

К основным отличиям Git от Subversion можно отнести следующее:

1. Автономная работа. SVN требует, чтобы участники были подключены к главному серверу хранилища, что существенно исключает работу в автономном режиме. Git же, наоборот, не требует постоянного подключения к серверу. Единожды склонировав хранилище можно полноценно работать, иногда, синхронизируясь с основным сервером, загружая в него свои изменения и сохраняя к себе изменения других разработчиков.
2. Разрешение конфликтов. Git также превосходит SVN, когда дело доходит до объединения и разрешения конфликтов. Git был разработан для работы с открытым исходным кодом, где многочисленные участники могут работать над одними и теми же частями кодовой базы. Для обеспечения такого типа совместной работы Git создала надежную систему разрешения конфликтов слияния, которая делает процесс более плавным и управляемым
3. Отказоустойчивость. Распределенная модель контроля версий Git помогает снизить вероятность потери основного хранилища. Поскольку участники клонируют основной репозиторий, риск полной потери вашего основного репозитория значительно снижается. С другой стороны, централизованная модель управления версиями SVN создает потенциал для единой точки отказа, если что-то случится с основным репозиторием.

Такая структура работы отлично подходит для Front-end разработки. Поэтому многие приложения, относящиеся к данной категории используют именно Git из-за его гибкости и большого числа возможностей.

### 1.2.3 Система управления проектами Jira

Текст

## 1.3 Нормативы описания разработок

## 1.3.1 Формат коммитов в системах контроля версий

Текст

## 1.3.2 Регламенты описания задач в системах управления проектами

Текст

## 1.4 Контроль качества продукта

### 1.4.1 Тестирование

Текст

### 1.4.2 Документация

Текст

## 1.4 Выводы к первой главе

Текст

# 2 Проектные решения разработки модуля системы csia для мониторинга описания работ по разработке по в системах контроля версий и управления проектами

## 2.1 Общие сведения о система CSIA

### 2.1.1 Платформа RadixWare

Текст

### 2.1.2 Модули системы CSIA

Текст

## 2.2 Требования к разрабатываемому модулю

Текст

## 2.3 Организация взаимодействия с системами контроля версий

## 2.3.1 Взаимодействие с Subverion

Текст

## 2.3.2 Взаимодействие с GitTea

Текст

## 2.4 Организация взаимодействия с системой управления проектами Jira

Текст

## 2.5 Мониторинг нарушений в проекте

### 2.5.1 Структурная модель

Текст

### 2.5.2 Алгоритм поиска нарушений

Текст

## 2.6 Генерация отчета

### 2.6.1 Работа со статистикой нарушений

Текст

### 2.6.2 Алгоритм генерации отчета

Текст

## 2.7 Выводы ко второй главе

Текст

# 3 Результаты опытной эксплуатации модуля системы csia для мониторинга описания работ по разработке по в системах контроля версий и управления проектами

## 3.1 Работа с разработанными сущностями

### 3.1.1 Сущность проекта

Текст

### 3.1.2 Сущность репозитория

Текст

### 3.1.3 Сущности системы управления проектами

Текст

## 3.2 Создание и настройка задачи для мониторинга

### 3.2.1 Задача Task.CheckProjectViolations

Текст

### 3.2.2 Настройка расписания

## 3.3 Пример сгенерированного отчёта

Текст

## 3.4 Выводы к третьей главе

Текст

# Заключение

Текст

# Список используемых источников

1 - <https://vk.com/doc7608079_449808456?hash=TrY4ho4utnq5LpCaxFpuamI4W8paaZPTR1sjBm5taBz&dl=49oI2tsG9r5KEMwqABnsociXSnqZQzVK3UstJ5VR2U4>

# Приложение А