УТВЕРЖДЕН
RU.29926343.62.01.29-01 31 1-1

СИСТЕМА ИНТЕЛЛЕКТУАЛЬНЫХ ПОМОЩНИКОВ КАДАСТР (СИП КАДАСТР)

Описание жизненного цикла и сопровождение продукта

RU.29926343.62.01.29-01 31 1-1

Листов 10

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Литера «»

RU.29926343.62.01.29-01 31 1-1

АННОТАЦИЯ

Система интеллектуальных помощников КАДАСТР (далее – СИП КАДАСТР), обеспечивает автоматизацию деятельности кадастровой оценки, является специальным сервисом-продуктом компании РЕД СОФТ. В рамках СИП КАДАСТР автоматизации подлежат процессы первичной кадастровой оценки с использованием технологий искусственного интеллекта.

В настоящем документе представлено описание системы управления жизненным циклом разработки и сопровождения СИП КАДАСТР, которая используется в РЕД СОФТ при разработке СИП КАДАСТР.

СОДЕРЖАНИЕ

1. ПО ДЛЯ ОБЕСПЕЧЕНИЯ ЖИЗНЕННОГО ЦИКЛА И СОПРОВОЖДЕНИЯ

Для обеспечения жизненного цикла разработки и сопровождения СИП КАДАСТР используется широкий ряд различного ПО.

1.1. При разработке СИП КАДАСТР для обеспечения контроля версий применяется программный продукт Git, который является свободно распространяемой по лицензии GNU/GPL v2 распределенной системой управления версиями. В РЕД СОФТ используется версия Git не ниже 2.0.

1.2. Для управления центральным репозитарием используется система GitLab, специальный инструмент жизненного цикла DevOps с открытым исходным кодом, представляющий систему управления репозиториями кода для Git. В текущий момент для разработки СИП КАДАСТР используется версия GitLab Community Edition 13.1.3.

1.3. Для создания версий для тестирования и релизов используется Docker, это специализованное программное обеспечение для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации. Для разработки СИП КАДАСТР используется версия Docker не ниже 17.

1.4. Для удобного управления несколькими контейнерами используется система оркестрации контейнеров Docker Compose. В основном используется самая актуальная версия системы.

1.1. Доступ к системе контроля версий

В РЕД СОФТ доступ к репозиториям системы Git и к системе GitLab обеспечивается из контролируемых зон объектов РЕД СОФТ удаленно с использованием шифруемых протоколов Secure Shell (SSH), HTTPS. Доступ разработчиков обеспечивается внутри корпоративной виртуальной частной сети (VPN), организуемой сертифицированными средствами межсетевого экранирования и построения VPN каналов.

1.2. Возможности системы контроля версий Git

Основными возможностями системы Git являются:

а) Возможность расширения функционала. Система спроектирована как набор программ, специально разработанных с учетом их использования в сценариях, это позволяет создавать специализированные системы контроля версий на базе Git и пользовательские интерфейсы;

б) Сохранение полной истории изменений файлов с возможностью возврата к предыдущим версиям. В истории сохраняются контрольные суммы изменений, поэтому она устойчива от внесения правок, а сам репозитарий представляет собой не сколько файлы, сколько изменения файлов;

в) Система поддерживает очень быстрое создание веток, это в том числе влияет на модель разработки продукта при использовании системы Git;

г) Синхронизация изменений между пользователями и автоматическое слияние изменений;

д) Репозитарий Git является распределенным, а потому устойчивым от потерь;

е) Возможность фиксации изменений в ветках. Возможность пометки определенных изменений тегами.

1.3. Возможности системы жизненного цикла GitLab

GitLab это интегрированный продукт, который автоматизирует весь жизненный цикл разработки программного обеспечения. GitLab расширяет возможности Git. Ключевые возможности GitLab:

а) Организация публичных и приватных репозиториев;
б) Облачный хостинг;
в) Неограниченное число пользователей, расширенное управление правами, группами;
г) Импорт проектов;
д) Вики-страницы;
е) Интеграция по API;
ж) Проставление лейблов и вех;
и) Использование поиска и шаблонов;
к) Комментирование, объединение;
л) Встроенные возможности для CI / CD автоматизации релизов — GitLab-CI;
м) Отслеживание изменений и прогресса;
н) Отслеживание времени;
п) Запросы на слияние кода;
р) Возможность проведения рецензирования кода.

1.4. Возможности системы автоматизации развертывания и управления приложениями Docker

Для ускорения и одновременно упрощения и улучшения качества продуктов в РЕД СОФТ используется система контейнеризации Docker. Эта система используется для развертывания СИП КАДАСТР в тестовой, разработки и в продуктовой средах.

Docker — это программная платформа для быстрой разработки, тестирования и развертывания приложений. Система койтенеризации Docker упаковывает ПО в стандартизованные блоки, которые называются контейнерами. Каждый контейнер включает все необходимое для работы приложения: библиотеки, системные инструменты, код и среду исполнения. Благодаря Docker можно быстро развертывать и масштабировать приложения в любой среде.

Ключевые возможности Docker:

а) Минимальное потребление ресурсов, по сравнению с виртуальной машиной. Контейнеры не виртуализируют всю операционную систему (ОС), а используют ядро хоста и изолируют программу на уровне процесса. Последний потребляет намного меньше ресурсов локального компьютера, чем виртуальная машина.

б) Быстрая доставка контейнеров/образов. Быстрое развертывание ПО. Легкая переносимость — приложения на основе контейнеров Docker можно эффективно и быстро переносить в разные системы, даже разные операционные системы.

в) Изоляция ПО. Процессы эффективно изолируются друг от друга, также ОС изолируется от контейнеров, что минимизирует риски нарушения безопасности и работоспособности инфраструктуры.

г) Простота работы. Упрощает задачи настройки, доставки, развертывания и масштабирования ПО. Повышает качество установки ПО и минимизирует человеческие ошибки.

д) Стандартизация операций. Все операции по работе с контейнерами и образами Docker стандартизированы.

е) Поддерживается системами непрерывной интеграции. В частности поддерживается системой GitLab-CI.

1.5. Возможности системы оркестрации Docker Compose

К основным возможностям системы оркестрации Docker Compose можно отнести:

а) Простота настройки и использования. Все настройки хранятся в отдельной папке с одним файлом конфигурации — docker-compose.yml. Управление сервисами: запуск, установка, настройка происходят простыми операциями и занимают минимальное время. Также возможно при этом организовывать сложные взаимосвязанные системы на базе нескольких приложений описанием в одном файле настроек.

б) Возможно создание нескольких изолированных сред на одном хосте, что также полезно при масштабировании систем.

2. МОДЕЛЬ РАБОТЫ

Общий процесс подготовки новых версий СИП КАДАСТР в целом соотвествует схеме (см. рис. 1):

2.1. Актуализация исходного кода (см. рис. 1 шаг 1) — осуществляется клонированием репозитария продукта или обновлением локального репозитария Git в основной ветке разработки (master). Этот этап осуществляется или консольным клиентом Git или любым другим клиентом Git.

2.2. Разработчик создает новую ветку от актуализированной основной ветки разработки для внесения изменений в исходный код продукта средствами клиента Git.

2.3. Разработчик вносит изменения в исходный код программы (см. рис. 1 шаг 2) любыми удобными для себя средствами разработки.

2.4. Разработчик фиксирует изменения и отправляет их в удаленный репозитарий Git на сервере GitLab (см. рис. 1 шаг 1), создавая при этом в удаленном репозитарии новую ветку — копию локальной ветки исходного кода, созданной в пункте п. 2.2.

2.5. Разработчик локально тестирует все изменения в версии программы.

2.6. Когда накопится достаточно изменений для новой версии в нескольких ветках или в одной новой ветке, данные изменения сливаются в главную ветку master. Также в случае критических исправлений все изменения поступают в главную ветку проекта. Без тага подготавливаемая в ветке master версия становится основным кандидатом в релиз.

2.7. Система GitLab-CI для проекта СИП КАДАСТР настроена таким образом, что при поступлении новых изменений в ветку master запускается автоматический пайплайн на сборку образа Docker новой версии ПО. Система СИП КАДАСТР состоит из нескольких репозитариев нескольких различных компонент. Так что слитие (merge) изменений в главную ветку (master) любого из проектов СИП КАДАСТР продуцирует автоматическое создание Docker образа с готовым ПО соотв. проекта (см. рис. 1 шаг 3).

2.8. В случае если произошла ошибка сборки в группу разработки проекта Telegram шлется уведомление об ошибке при сборке проекта.

2.9. Успешно созданные Docker образы компонент системы СИП КАДАСТР автоматически регистрируются в репозитарий образов Docker (см. рис. 1 шаг 4). Новые Docker образы компонент системы СИП КАДАСТР регистрируются в репозитарии сборки Docker под тагами master (по имени ветки для которой строился проект).

2.10. В тестовой среде происходит настройка запуска готового проекта СИП КАДАСТР (см. рис. 1 шаг 5). Для этого из репозитария образов Docker выкачиваются последние (актуальные) версии (версии помеченные тагом master) всех компонент системы СИП КАДАСТР. Делается это командой docker pull в ручном режиме.

2.11. Тестовая среда СИП КАДАСТР запускается посредством ранее настроенной системы оркестрации Docker Compose.

2.12. Тестовое приложение СИП КАДАСТР, состоящее из нескольких компонент, и они же кандидаты на новые версии тестируется (см. рис. 1 шаг 5). В случае выявленных проблем все шаги повторяются сначала для устранения ошибок и доработки функционала.

2.13. В случае если приложение готово к созданию версии, все ошибки устранены, все требования к новой версии выполнены: все репозитарии ветки master всех компонент системы СИП КАДАСТР фиксируются тагом с номером версии в формате «vM.m.B», где M — главный номер версии, m — младший номер версии, B — «build», используется для указания минимальных изменений в проекте. Версии компонент повышаются.

2.14. Система GitLab-CI для проекта СИП КАДАСТР настроена таким образом, что при фиксации изменений в тагах запускается автоматический пайплайн на сборку образа Docker

новой версии, но уже релиза с именем версии указанной в таге шаги 3-4 повторяются (см. рис. 1). В репозитарии образов Docker фиксируются новые версии образов Docker отмеченные тагом в формате «vM.m.B». Это релизные сборки компонент системы СИП КАДАСТР и кандидаты на установку в продуктовую среду.

2.15. Данные образы поступают на установку в продуктивную среду выполнения системы СИП КАДАСТР (см. рис. 1 шаг 6), в ручном режиме, аналогично тому как настраивается и запускается тестовая среда см. п. 2.11.

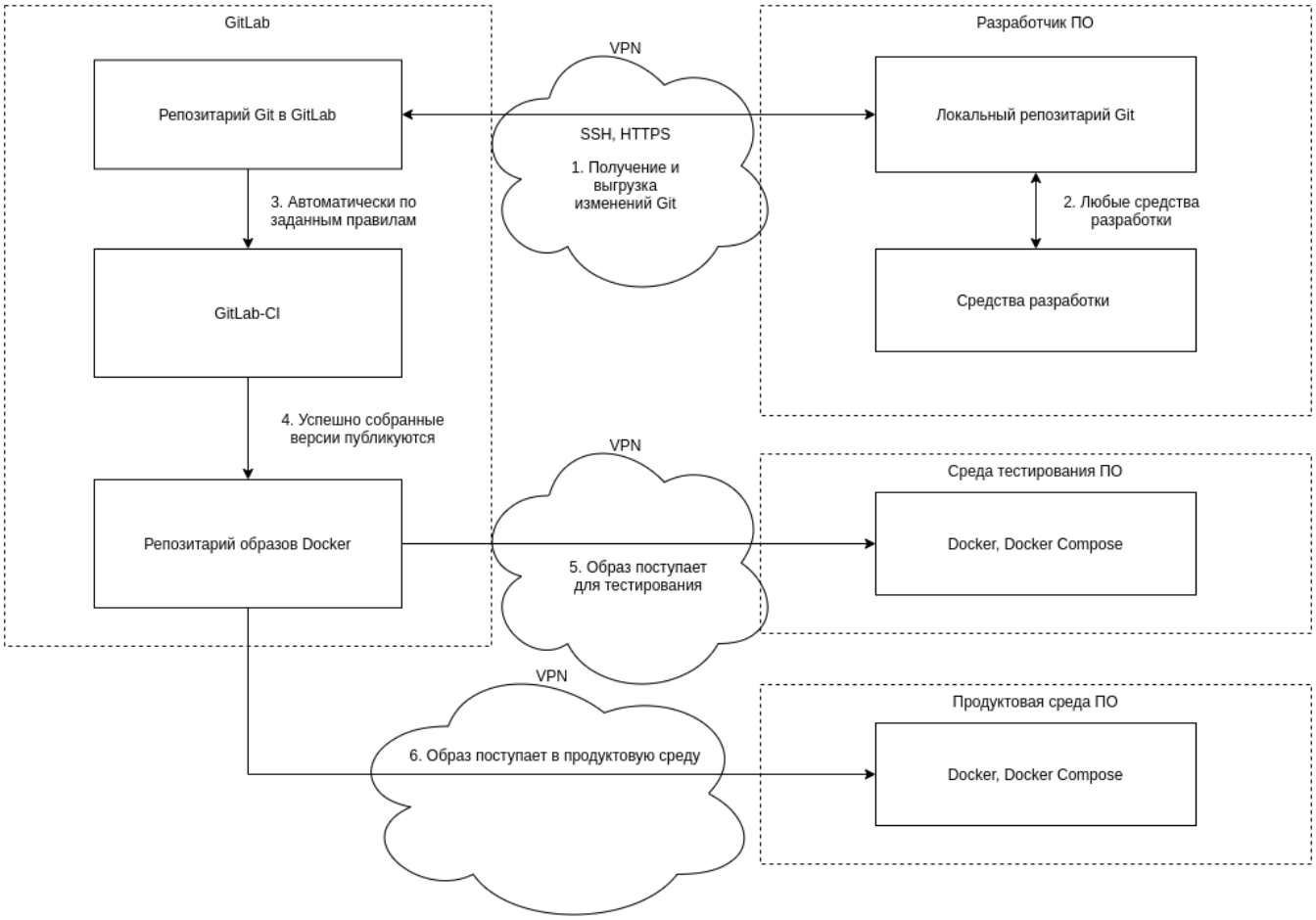

Рисунок 1

3. СИСТЕМА ОТСЛЕЖИВАНИЯ ОШИБОК

При разработке СИП КАДАСТР используется система GitLab для управления задачами. Также эта система используется для приема заявок на доработку и для регистрации ошибок.

Ключевые возможности системы управления задач GitLab являются:

а) Произвольное число канбан-досок с задачами с разделением по проектам;

б) Просмотр задач списком;

в) Отметка задач тагами;

г) Статусы задач;

д) Возможность вести обсуждение по каждой задаче;

е) Учет времени.

3.1. Жизненный цикл задач

В применяемой при разработке СИП КАДАСТР используется следующая схема жизненного цикла задач:

3.1.1. Регистрация ошибки или требований к доработке системы в задаче GitLab. Прием заявок на доработку, а также регистрация ошибок организована через каналы Telegram.

3.1.2. Далее задачи рапределяются по исполнителям, а также им назначается при необходимости срок выполнения.

3.1.3. В момент когда разработчик приступает к выполнению задачи, задача переводится со статуса «Открытый» в статус «В работе».

3.1.4. При завершении задачи задача переводится со статуса «В работе» на статус «На проверке». Тогда же задача проверяется тестировщиком или аналитиком или заказчиком на тестовом стенде.

3.1.5. В случае успешного выполнения задачи она переводится на статус «Закрыт». Если же при проверке задачи были выявлены ошибки, задача вновь переводится на статус «В работе».

ПЕРЕЧЕНЬ ТЕРМИНОВ

Термин	Определение
1. **API**	программный интерфейс приложения, интерфейс прикладного программирования.
2. **CI / CD**	практика разработки программного обеспечения, которая заключается в постоянном слиянии рабочих копий в общую основную ветвь разработки (до нескольких раз в день) и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем.
3. **DevOps**	методология активного взаимодействия специалистов по разработке со специалистами по информационно-технологическому обслуживанию и взаимная интеграция их рабочих процессов друг в друга для обеспечения качества продукта.
4. **Docker**	программное обеспечение для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесен на любую Linux-систему с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами.
5. **Git**	распределённая система управления версиями.
6. **GitLab**	веб-инструмент жизненного цикла DevOps с открытым исходным кодом, представляющий систему управления репозиториями кода для Git с собственной вики, системой отслеживания ошибок, CI/CD пайплайном и другими функциями.
7. **HTTPS**	расширение протокола HTTP для поддержки шифрования в целях повышения безопасности. Данные в протоколе HTTPS передаются поверх криптографических протоколов TLS или устаревшего в 2015 году SSL.
8. **Secure Shell (SSH)**	сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов).
9. **Telegram**	кроссплатформенный мессенджер, позволяющий обмениваться сообщениями и медиафайлами многих форматов.
10. **VPN**	обобщённое название технологий, позволяющих обеспечить одно или несколько сетевых соединений (логическую сеть) поверх другой сети (например Интернет).
11. **Канбан**	система организации производства и снабжения, позволяющая реализовать принцип «точно в срок».

Термин	Определение
12. **Канбан-доска**	является одним из инструментов, который может использоваться при внедрении метода управления разработкой «канбан».
13. **Контейнеризация**	(иначе, виртуализация на уровне операционной системы, контейнерная виртуализация, зонная виртуализация) метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного. Эти экземпляры (обычно называемые контейнерами или зонами) с точки зрения пользователя полностью идентичны отдельному экземпляру операционной системы.
14. **ОС**	комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.
15. **Облачный хостинг**	тип веб-хостинга, который использует несколько разных серверов для того, чтобы распределять нагрузку и максимально увеличивать время безотказной работы.
16. **ПО**	программа или множество программ, используемых для управления компьютером (ISO/IEC 26514:2008).
17. **Продуктовая среда**	конкретный экземпляр конфигурации аппаратного и программного обеспечения, предназначенный для работы в контролируемой среде. В данной среде продукт доступен пользователям.
18. **РЕД СОФТ**	отечественный поставщик решений и услуг в области информационных технологий.
19. **Рецензирование кода**	(иначе обзор кода, ревизия кода (англ. code review) или инспекция кода (англ. code inspection) — систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки. Целью обзора является улучшение качества программного продукта и совершенствование навыков разработчика.
20. **Среда разработки**	интегрированная система, включающая в себя аппаратные средства, ПО, программно аппаратные средства, процедуры и документы, необходимые для разработки ПО.
21. **Тестовая среда**	конкретный экземпляр конфигурации аппаратного и программного обеспечения, предназначенный для тестирования работы в контролируемой среде.