ОПИСАНИЕ ПРОЦЕССОВ, ОБЕСПЕЧИВАЮЩИХ ПОДДЕРЖАНИЕ ЖИЗНЕННОГО ЦИКЛА

СИСТЕМЫ BERGEN INTEGRATION SUITE (BIS)

Листов 19

2021

Оглавление

Термины и сокращения

Сокращение	Полное наименование
ЦОД	Центр обработки данных
JavaScript	Высокоуровневый язык программирования
HTML	Стандартизированный язык разметки веб-страниц
CSS	Формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML)
Си	Компилируемый статически типизированный язык программирования общего назначения
Java	Объектно-ориентированный язык программирования
Vuejs-фреймворк	Фреймворк для приложений, написанный на языке программирования JavaScript
OpenJDK	Программная платформа, которая подходит для разных языков программирования
Linux	Семейство Unix-подобных операционных систем на базе ядра Linux
RM	RedMine – система управления проектом
Intellij IDEA	Редактор исходного кода для кроссплатформенной разработки веб- и облачных приложений
Vim	Свободный текстовый редактор с полной свободой настройки и автоматизации
Flyway	Инструмент миграций и версионности изменений схемы СУБД.
Docker	Программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации
Artemis MQ	Брокер сообщений с открытым исходным кодом реализующий спецификацию JMS и AMQP.

Сокращение	Полное наименование
SQL	Язык программирования структурированных запросов
PostgreSQL	Свободная объектно-реляционная система управления базами данных
Git	Распределенная система управления версиями
Code-review	Систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки
АДМ	Системный администратор проекта
АРП	Архитектор проекта
БД	База данных
БП	Бизнес–процесс
ОС	Операционная система
ПП	Программный продукт
СУБД	Система управления базами данных
Система	Системный комплекс Bergen Integration Suite
IDE	Интегрированная среда разработки, система программных средств, используемая программистами для разработки программного обеспечения

1. Жизненный цикл процессов реализации (разработки) Системы

1.1. Ландшафт ведения разработки

Ландшафт системы разработки представлен в виде следующих сред, которые на практике являются отдельным kubernetes кластерами:

a) DEV (сервер разработки);

b) QA (сервер тестирования);

c) PROD (продуктивные сервер).

Процесс работы выглядит так:

a) Разработчики работают над новой feature (задачей) и осуществляют отладку на локальной машине или DEV среде;

b) Разработчики проверяют и тестируют готовые feature (задачи), собранные в DEV репозитории, в QA среде;

c) Разработчики выпускают новый релиз, и ответственный делает миграцию успешного релиза в окружение PROD заказчика(-ов).

1.2. Требования к оформлению программы

Программа разрабатывается в средах:

a) Intellij IDEA;

b) Vim.

Настройки лежат в репозиториях проекта Bergen BIS.

Перечень репозиториев проекта Bergen BIS представлен ниже, см. Таблица 1 – Список репозиториев.

Таблица 1 – Список репозиториев

Репозиторий в Git	Ветки	QA сервер
https://gitlab.bergen.tech/c_mvp/bis-console	dev, master	https://stable-today.qa.bis.bergen.tech/
https://gitlab.bergen.tech/c_mvp/bis-api-gateway	dev, master	
https://gitlab.bergen.tech/c_mvp/bis-addressing-service	dev, master	
https://gitlab.bergen.tech/c_mvp/bis-data-presenter	dev, master	
https://gitlab.bergen.tech/c_mvp/bis-data-presenter	dev, master	
https://gitlab.bergen.tech/c_mvp/bis-docker-engine	dev, master	
https://gitlab.bergen.tech/c_mvp/bis-logging	dev, master	
https://gitlab.bergen.tech/c_mvp/bis-mertics-exporter	dev, master	
https://gitlab.bergen.tech/c_mvp/bis-monitoring	dev, master	
https://gitlab.bergen.tech/c_mvp/bis-state-aggregator	dev, master	

1.3.Регламент ведения разработки и тестирования

a) Git-репозитории разделены логически по компонентам Системы;

b) Каждый репозиторий един;

c) В git-репозиториях должны отсутствовать:

- Генерируемые файлы и результаты сборки, включая документацию, логи, upload файлы;
- node_modules и другие скачиваемые библиотеки;
- Любые пароли, сертификаты, API ключи;
- Настройки локального окружения, IDE.

d) Для документации по всем репозиториям создаётся отдельный репозиторий, который обновляется при необходимости. Каждый метод должен быть задокументирован в системе документации (java docs, open api или иной);

e) Пароли, сертификаты и ключи доступа могут быть сохранены в «личном» репозитории только при шифровании с помощью публичного ключа соответствующего получателя;

f) По умолчанию для проверки оформления кода на соответствие проектным стандартам запускается linter;

g) Разработчик отвечает за отсутствие merge-конфликтов при слиянии в dev;

h) Контроль технической документации и код-ревью перед переносом в master осуществляет руководитель разработки;

– MR (merge request) в DEV ветку создает разработчик, выполняющий задачу. Разработчик, который создал MR должен отправить ссылку на MR проверяющему. MR обрабатываются по порядку в сторону увеличения номера. При наличии замечаний MR отклоняется. Задача переводится на сотрудника, который делает MR.

1.4. Работа с Git

Работа с репозиторием производится в соответствии с регламентом, описанным в руководстве разработчика:

a) master: в данной ветке хранится продакшн-версия проекта, при этом изменения в этой ветке не производятся напрямую, а вливаются из dev ветки;

b) dev: в данной ветке интегрируются изменения, внесённые всеми разработчиками. При этом разработчики не могут вносить изменения непосредственно в этой ветке, а должны переносить в неё изменения из веток feature с помощью merge request;

c) release/release-x.x.x: как только dev ветка набрала в себя достаточно изменений и релиз менеджер считает, что пора выпустить новую версию, необходимо сделать ответвление от dev в release. С этого момента, в release ветку больше нельзя добавлять никаких изменений функционала, исключение составляют только срочная правка ошибок

(fix ветки). Как только release бранч готов к эксплуатации, ему присваивается версия, и через merge request он сливается с master веткой;

d) feature/<name>: каждая такая ветка создаётся отдельным разработчиком при добавлении функционала. Одна задача - одна ветка. Не допускать выполнения многих задач в одной ветке, так как это затруднит последующий merge. Разработчик вносит изменения в код только на ветке feature. После выполнения работ на данной ветке разработчик переносит изменения в dev через merge request. При выполнении большой задачи в течении долгого времени обязательно каждые 1-2 дня необходимо делать merge request в основную ветку разработки, чтобы избежать накопления большой разницы в коде и тяжело решаемых конфликтов изменений;

e) fix/<name>: ветки для внесения исправлений в release ветки. Могут создаваться только от release веток. После исправления вливается через merge request в ту же ветку, от которой была создана и дополнительно в master и dev ветку, чтобы актуальный репозиторий содержал в себе все исправления.

При создании merge request необходимо выбрать ветку, в которой шла разработка и целевую ветку, см. Рисунок 1:

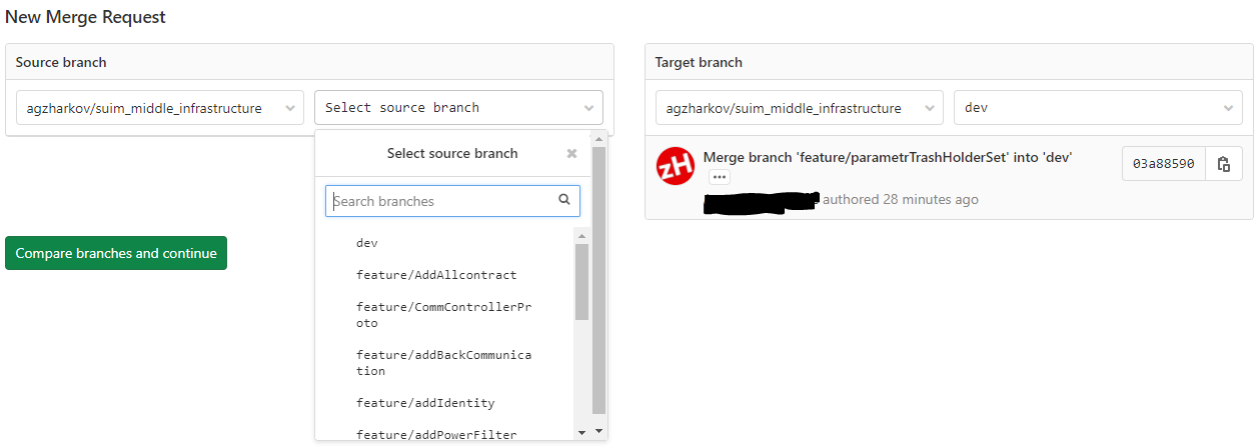

Рисунок 1 – Выбор ветки разработки

В настройках необходимо указать Approvals. Список людей, кто будет подтверждать merge. Это должен быть человек с ролью АРП. Если необходимо, то выставляем галочку для удаления ветки разработки, после подтверждения слияния, см. Рисунок 2:

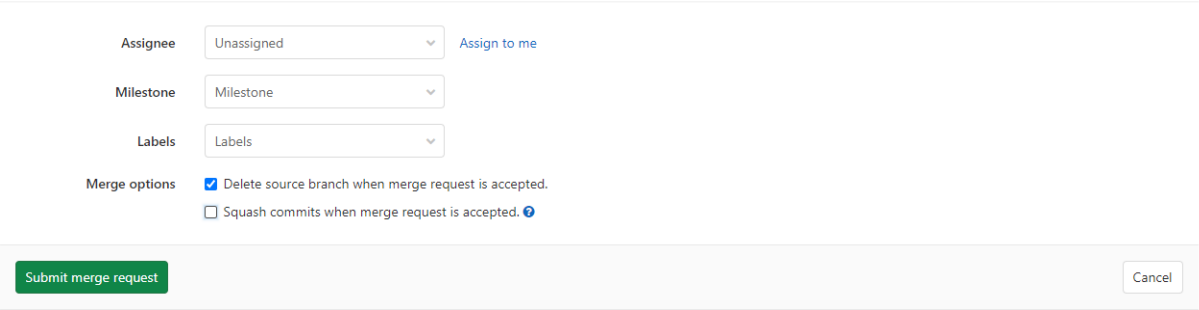

Рисунок 2 – Простановка чекбокса удаления ветки

Правила именования веток: feature/[название разработки]

Правила именования коммитов: в наименовании коммита и веток запрещено писать комментарии, не относящиеся к проекту.

1.5.Сборка ПО

Описание и состав сборки ПО, см. Таблица 2.

Таблица 2 – Описание и состав сборки ПО

Имя файла	Описание
bin	Исполняемые sh скрипты уровня старта приложения docker entrypoint.sh и т.д..
chart	helm chart приложения.
app	Исходный код приложения. В зависимости от типа приложения имеет специфичную структуру для данного приложения.
.envvars	Список переменных окружения.
Dockerfile	Файл описания сборки docker image для приложения.
.ci-env	Список переменных окружения необходимый для CI/CD gitlab.
README.md	Описание приложения, расположенного в репозитории, техническая информация для

Имя файла	Описание
	разработчика, описания и краткие инструкции.
.gitlab-ci.yml	Файл pipline этапов для CI/CD gitlab описывающий проверку, сборку, тестирование и релиз приложения.

1.6.Порядок выпуска релизов

Релиз формируется по окончании спринта, когда завершено ручное тестирование на функционал, не покрытый автоматическими тестами, а также успешно пройдены автоматические тесты, функциональные, интеграционные, модульные тесты, тесты, проверяющие инсталляцию дистрибутива Системы, нагрузочные тесты.

a) **DEV Kubernetes кластер**

Временный kubernetes кластер может быть развернут под определенную ветку или коммит содержащий необходимый набор модулей требуемых версий. Для проверки разработчиком. Разворачивается Система средствами CI/CD gitlab.

b) **QA Kubernetes кластер**

Обновляется по мере накопления стабильных изменений и(или) критический исправлений на DEV сервере. В конце итерации разработки длительностью от 1 до 2 недель принимается решение о выпуске релиза.

2. Поддержание жизненного цикла Системы

Поддержание жизненного цикла Системы осуществляется за счет сопровождения Системы и включает в себя проведение модернизаций Системы в соответствии с собственным планом доработок и по заявкам клиентов, консультации по вопросам установки и эксплуатации (по электронной почте) Системы.

В рамках технической поддержки Системы оказываются следующие услуги:

– Помощь в установке Системы;

– Помощь в настройке и администрировании;

– Помощь в установке обновлений Системы;

– Помощь в поиске и устранении проблем в случае некорректной установки обновления Системы;

– Пояснение функционала модулей Системы, помощь в эксплуатации Системы.

2.1 Данные о персонале, задействованном в процессе разработки и тестирования

В процессе разработки и тестирования задействовано 17 штатных сотрудников. Данные о персонале, задействованном в процессе разработки и тестирования, представлены в таблице ниже, см. Таблица 3:

Таблица 3 – Данные о персонале, задействованном в процессе разработки и тестирования

Отдел	Должность
Администрация	Директор по консалтингу
Администрация	Системный администратор
Администрация	Администратор проекта
Администрация	Дизайнер
Администрация	Специалист по кадрам
Администрация	Руководитель проекта
Отдел бухгалтерского учёта	Главный бухгалтер
Отдел разработки программного обеспечения	Младший разработчик 2 уровня
Отдел разработки программного обеспечения	Младший разработчик 3 уровня
Отдел разработки программного обеспечения	Специалист DevOps
Отдел разработки программного обеспечения	Старший разработчик 2 уровня
Отдел разработки программного обеспечения	Старший разработчик 5 уровня
Отдел разработки программного обеспечения	Старший разработчик 5 уровня
Отдел бизнес и системной аналитики	Аналитик 5 уровня
Отдел бизнес и системной аналитики	Тестировщик 1 уровня
Отдел бизнес и системной аналитики	Тестировщик 1 уровня

Отдел	Должность
Отдел бизнес и системной аналитики	Руководитель отдела

С точки зрения разработки в процессе создания, развития и ввода в эксплуатацию компонентов системы участвуют следующие роли, см. Таблица 4

Таблица 4 – Роли и зоны ответственности ролей

Роль	Зона ответственности
Руководитель проекта	Ответственный за организацию работ в рамках проектной задачи.
IT-лидер	Ответственный за техническую и процессную составляющую разработки Системы.
Администратор (АДМ)	Ответственный за организацию и поддержку ландшафта системы, ведение бэкапов, ведение прав для разработчиков, настройку интеграции с внешними системами.
	Ответственный за сборку релиза и заливку на QA и PROD.
	Ответственный за организацию обеспечения ИБ.
Руководитель разработки (Архитектор (АРП))	Ответственный за техническую архитектуру разрабатываемой Системы.
Разработчик	Ответственный за разработку в рамках назначенных задач.
Тестировщик	Ответственный за тестирование и качество разрабатываемой Системы.
Специалист службы поддержки	Ответственный за принятие заявок от заказчика и консультирование Пользователей Системы.

Фактический адрес, по которому осуществляется процесс разработки и тестирования, заявляемого ПО:

Россия: г. Москва, ул. Мясницкая, дом 38, стр. 1.

2.2 Информация о процессе сопровождения и поддержки

Данные о возможных средствах коммуникации и о режиме работы службы поддержки:

- Коммуникация со службой поддержки ведется по электронной почте по адресу: support@bergen.tech или по телефону 8 (495) 664-37-83;
- Служба поддержки работает по будним дням с 10:00 до 19:00 по МСК;

В процессе сопровождения задействовано 4 штатных сотрудника. Данные о персонале, задействованном в процессе сопровождения, представлены в таблице ниже, см. Таблица 5.

Таблица 5 – Данные о персонале, задействованном в процессе сопровождения

Отдел	Должность
Администрация	Системный администратор
Отдел бизнес и системной аналитики	Ведущий аналитик-исследователь
Отдел разработки программного обеспечения	Старший специалист разработчик 5 уровня
Отдел техподдержки	Ведущий специалист службы поддержки 4 уровня

Фактический адрес, по которому осуществляется процесс сопровождения: Россия: г. Москва, ул. Мясницкая, дом 38, стр. 1.

Для работы с Системой пользователю необходимо ознакомиться с Руководством пользователя.

Пользователи Системы должны обладать навыками работы с персональным компьютером и уметь пользоваться браузером.

3. Устранение неисправностей, выявленных в ходе эксплуатации Системы

Неисправности, выявленные в ходе эксплуатации Системы, могут быть исправлены двумя способами:

- Массовое плановое обновление компонентов Системы;
- Единичная работа специалиста службы технической поддержки по запросу пользователя.

В случае возникновения неисправностей в Системе, или необходимости в её доработке, Заказчик направляет Разработчику запрос. Запрос должен содержать тему запроса, суть (описание) и по мере возможности снимок экрана со сбоем (если имеется сбой).

Запросы могут быть следующего вида:

– Наличие Инцидента – произошедший сбой в системе у одного Пользователя со стороны Заказчика;
– Наличие Проблемы – сбой, повлекший за собой остановку работы/потерю работоспособности Программы;
– Запрос на обслуживание – запрос на предоставление информации;
– Запрос на развитие – запрос на проведение доработок Программы.

Запрос направляется Заказчиком либо Пользователями Заказчика через запрос в службу поддержки по электронной почте на электронный адрес tech

Пример:

– Пользователь Заказчика обнаружил проблему и зафиксировал ее письмом на службу поддержки Системы;
– Служба поддержки создает в системе управления проектом RedMine задачу с номером №100000 и названием «Доработка backend части мониторинга событий»; указывается критичность задачи исходя из влияния проблемы на работоспособность Системы; задача назначается на руководителя разработки, имеющего проектную роль Архитектора проекта (АРП); статус задачи «Открытая»;
– Руководитель разработки назначает задачу №100000 на аналитика для уточнения постановки или дополняет постановку самостоятельно и назначает задачу сразу на разработчика;
– Когда разработчик приступает к задаче, то должен изменить статус задачи на «В Работе» и создать ветку от dev с названием «feature/<name>-

100000», в котором он будет вести свою работу. В данном примере <name> задается семантически по смыслу задачи;

– После окончания разработки и локального тестирования, и проверки на DEV сервере разработчик должен создать merge request и влить изменения в ветку dev. Merge request подтверждает сам разработчик. Текст коммита должен соответствовать следующему формату:

  - «#NNNNN <Описание изменений>», где NNNNN – номер задачи в RedMine;
  - В тексте коммита желательно придерживаться обезличенных формулировок. То есть, «исправлена проблема, связанная с ….», «добавлен функционал, позволяющий …..»;

– Далее разработчик или администратор (по запросу разработчика) запускает pipeline для разворачивания задачи на QA сервере, и разработчик переводит задачу в статус «В тесте» и назначает на ответственного тестировщика. Статус задачи «В тесте» означает, что разработка закончена, локально протестирована, код смержен в dev ветку и развернут на QA сервере;

– Тестировщик проверяет задачу на QA сервере и если тест пройден, то ставит статус «В релиз» и назначает на Администратора (АДМ). Если тест не пройдет, возвращает разработчику, ставит статус «На доработке» и оставляет комментарий с описанием проблемы;

– После формирования релиза Администратор переводит задачи из статуса «В релиз» в статус «Решена». То есть статус «Решена» означает, что код находится в release ветке, был успешно протестирован на QA сервере и готова к сборке релиза;

– «Решена» – код находится в release ветке, успешно был протестирован на QA сервере и готова к сборке релиза;

– Администратор Системы по договоренности с Заказчиком обновляет Систему Заказчика.

4. Совершенствование Системы

Система регулярно развивается: в ней появляются новые дополнительные возможности, оптимизируется нагрузка ресурсов ПК, серверов, обновляется интерфейс.

В процессе совершенствования задействовано 5 штатных сотрудника. Данные о персонале, задействованном в процессе совершенствования, представлены в таблице ниже, см. Таблица 6.

Таблица 6 – Данные о персонале, задействованном в процессе совершенствования

Отдел	Должность
Администрация	Системный администратор
Отдел бизнес и системной аналитики	Ведущий аналитик-исследователь
Отдел разработки программного обеспечения	Старший специалист разработчик 2 уровня
Отдел разработки программного обеспечения	Старший специалист разработчик 5 уровня
Отдел разработки программного обеспечения	Старший специалист разработчик 5 уровня

Фактический адрес, по которому осуществляется процесс совершенствования: Россия: г. Москва, ул. Мясницкая, дом 38, стр. 1.

Пользователь может самостоятельно повлиять на совершенствование продукта, для этого необходимо направить предложение по усовершенствованию на электронную почту технической поддержки по адресу tech.

Предложение будет рассмотрено и, в случае признания его эффективности, в Систему будут внесены соответствующие изменения.

5. Техническая поддержка Системы

Для оказания технической поддержки Системы, Пользователи Системы могут направлять возникающие вопросы на электронную почту технической

поддержки по адресу support@bergen.tech или по телефону 8 (495) 664-37-83. Техническая поддержка осуществляется в будние дни с 10:00 до 19:00 по МСК.

5.1 Данные о персонале, задействованном в процессе техподдержки

В процессе техподдержки задействовано 3 штатных сотрудника. Данные о персонале, задействованном в процессе техподдержки, представлены в таблице ниже, см. Таблица 7.

Таблица 7 – Данные о персонале, задействованном в процессе техподдержки

Отдел	Должность
Администрация	Системный администратор
Отдел разработки программного обеспечения	Специалист DevOps
Отдел разработки программного обеспечения	Старший разработчик 2 уровня

Фактический адрес, по которому осуществляется процесс техподдержки: Россия: г. Москва, ул. Мясницкая, дом 38, стр. 1.

Лист регистрации изменений									
Изм.	Номера листов (страниц)				Всего листов (стр.в докум.)	№ докумен-та	Входящий № сопро-водительного документа и дата	Под-пись	Да-та
	изме-ненных	заме-ненных	новых	аннули-рованных					

Составили

Версия	Дата	Фамилия, имя, отчество	Должность исполнителя	Подпись
0.1	28.02.2021	Мишкина А.А.	Аналитик	
0.2	15.03.2021	Мишкина А.А.	Аналитик	
0.3	26.03.2021	Мишкина А.А.	Аналитик	
1.1	31.03.2021	Мишкина А.А.	Аналитик	
1.2	07.04.2021	Харитонов А.Н.	Аналитик	