РУКОВОДСТВО РАЗРАБОТЧИКА

на

СИСТЕМУ BERGEN INTELLIGENT PROCESS AUTOMATION

25 Листов

2021

Аннотация

Настоящий документ содержит описание общих принципов разработки системы Bergen IPA.

Настоящий документ является составной частью комплекта рабочей документации на Систему и разработан в соответствии с требованиями документа «РД 50-34-698-90 Комплекс стандартов и руководящих документов на автоматизированные системы» и ГОСТ 2.105-95 «Единая система конструкторской документации. Общие требования к текстовым документам».

Содержание

Введение

Bergen IPA - Универсальная модульная платформа с элементами машинного обучения для автоматизации как простых бизнес-процессов уровня операционной деятельности предприятия, так и сложных, в рамках которых необходим специализированный анализ данных для принятия оптимальных решений.

Перед началом работы команда разработчиков Системы должна ознакомиться с текущим документом.

1. Общая информация

1.1 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.
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

Сокращение	Полное наименование
АРП	Архитектор проекта
БД	База данных
БП	Бизнес–процесс
ОС	Операционная система
ПП	Программный продукт
СУБД	Система управления базами данных
Система	Системный комплекс Bergen Intelligent Process Automation
IDE	Интегрированная среда разработки, система программных средств, используемая программистами для разработки программного обеспечения

1.2 Область применения

Система может применяться в любой области, где необходимо:

- Гибкая настройка БП Контроль выполнения БП на каждом этапе
- Контроль выполнения БП на каждом этапе
- Встроенная поддержка технологий машинного обучения
- Повышение прозрачности бизнеса
- Взаимодействие между подразделениями и отделами становится более понятным за счёт определения ответственных лиц для каждого этапа, описания вариантов исполнения процесса в зависимости от входных данных, повышения прозрачности и гибкости в принятии решений и полученного результата.

Примеры возможного применения:

- Автоматизация как простых бизнес-процессов уровня операционной деятельности предприятия, так и сложных, в рамках которых необходим специализированный анализ данных для принятия решений.
- Гибкая настройка бизнес-процессов.
- Контроль выполнения бизнес-процессов на каждом этапе.
- Подключение в виде плагинов различных внешних платформ и сервисов
- Принятие оптимальных решений при помощи машинного обучения
- Построение модели при помощи машинного обучения с учителем
- Упрощенное взаимодействие между подразделениями и отделами за счёт определения ответственных лиц для каждого этапа, описания вариантов исполнения процесса в зависимости от входных данных,

повышения прозрачности и гибкости в принятии решений и полученного результата.

1.3 Требования к подготовке разработчиков

Разработка и развитие (в том числе модернизация) компонентов Системы осуществляется специалистами Bergen IT.

Для успешного выполнения работ по разработке системы:

Разработчики должны обладать следующими обязательными знаниями:

a) Опыт работы с:

- Visual Code;
- Docker;

b) Уверенное знание принципов работы систем контроля версий (Git);

Также для выполнения работ по разработке Системы разработчики должны соответствовать следующим рекомендуемым требованиям:

c) Опыт работы: OS Linux, JavaScript, HTML, CSS, фреймворк Vue.js, Vuetify, Java, Golang, gRPC, REST API, Docker, Artemis MQ или другой брокер сообщений, SQL, PostgreSQL или другая реляционная БД, желателен опыт Docker;

d) Базовые знания в области автоматизации процесса разработки (CI/CD) с применением ПП Gitlab.

Руководитель разработки должен обладать перечисленными выше знаниями, а также:

a) Опыт разработки ПО, не менее четырех лет;
b) Опыт руководства группой разработчиков, не менее одного года;
c) Знание положений об обеспечении информационной безопасности;
d) Понимание технических решений Системы в объеме технической документации на решения;
e) Опыт применения различных методик управления исходным кодом: code-review, merge-request, feature-branch, регулярная поставка релизов и т.п.

Разработчики должны соблюдать требования нормативных и иных актов в области информационной безопасности принятых в Компании.

1.4 Перечень документов, обязательных для ознакомления разработчиками перед выполнением работ

Для разработки Системы разработчику необходимо ознакомиться со следующими документами и стандартами разработки:

a) Положениями настоящего документа;

b) Руководством пользователя Системы.

2 Краткое описание Системы

2.1 Общая архитектура

Система представляет собой универсальную платформу интеллектуальной автоматизации процессов.

Система может решать следующие задачи, относящиеся к классу задач систем класса:

1. Выстраивание и автоматизация бизнес-процессов с нуля, поддержка уже существующих, в том числе с поэтапным переходом, расширением зоны охвата и адаптацией по мере необходимости.
2. Предоставление инновационного решения с элементами машинного обучения для автоматизации бизнес- процессов, в рамках которых необходим специализированный анализ данных для принятия решений.
3. Предоставление модульного подхода для автоматизации бизнес-процессов, позволяющего на этапах выполнения бизнес-задач подключать в виде плагинов различные внешние платформы и сервисы.

Программа относится к следующим классам приложений:

1. Системы управления проектами, исследованиями, разработкой, проектированием и внедрением
2. Приложения общие для повышения эффективности бизнеса и приложения для домашнего; пользования, отдельно реализуемые;
3. Серверное и связующее программное.

Программа предназначена для решения следующих задач:

- Выстраивание и автоматизация бизнес-процессов с нуля, поддержка уже существующих, в том числе с поэтапным переходом, расширением зоны охвата и адаптацией по мере необходимости;
- Использование элементов машинного обучения для автоматизации бизнес-процессов, в рамках которых необходим специализированный анализ данных для принятия оптимальных решений.

2.2 Компоненты Bergen IPA

В системе Bergen IPA можно выделить следующие компоненты:

- Vue UI приложение
- APIGW

- PlatformaAPI
- Camunda
- FileUpload
- Minio
- KeyCloak
- PluginsEngine
- PostgresDB
- RoleService
- GitLoader

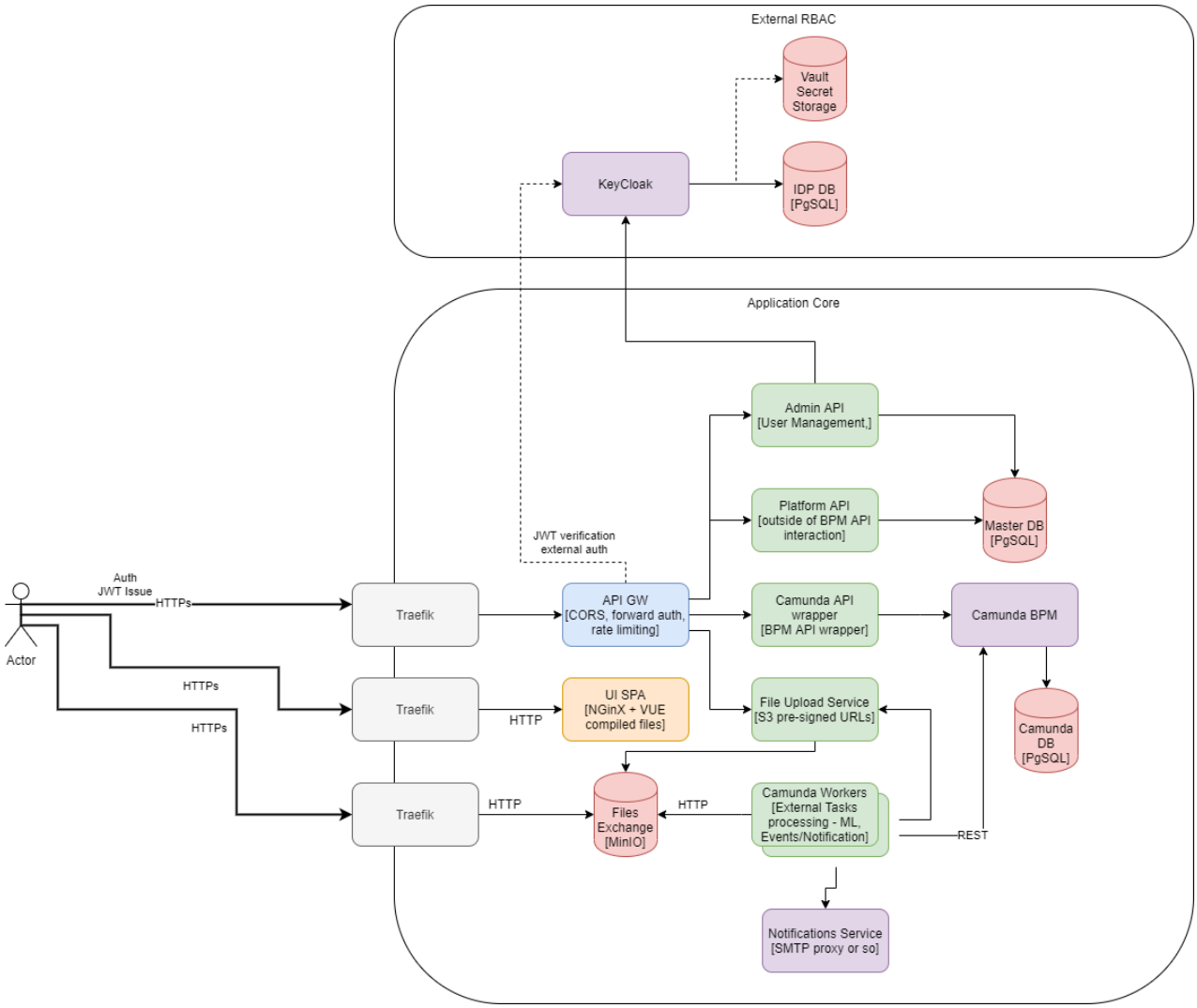


Рисунок 1 – Компоненты Bergen IPA.

Пользовательский интерфейс Console

Пользовательский интерфейс позволяет осуществлять работу с загруженным БП через браузер

- Admin API

Администрирование пользователей, групп, и их полномочий.

- UI SPA[NGinX+VUE compiled files]

Веб интерфейс пользователя, реализует весь пользовательский функционал платформы.

– KeyCloak

Единый сервис аутентификации пользователя. Выдает JWT токен для Vue приложения.

– APIGW

Входная точка для взаимодействия с функционалом платформы.

- Реализуется функционал проверки токена пользователя посредством его отправки на KeyCloak.
- Выполняет роль маршрутизатора на прочие сервисы системы.
- Выполняет функции коммутации по HTTP с Camunda. Запросы отправляются напрямую в прокси режиме за исключением случая загрузки BMPN схемы из git. В этом случае реализована дополнительная связка с сервисом GitLoader.
- Выполняет функции коммутации по GRPC со всеми прочими сервисами. Соединение реализуется через технологию grpc-gateway

– PlatformaAPI

Сервис функционала управления платформой. Функционал платформы расширяет собой функционал Camunda.

- Реализует функционал управления функциональными модулям
- Реализует функционал управления экранными формами
- Планируется дальнейшие расширение

– Camunda

Ядро платформы, движок bpmn бизнес-процесса. Предоставляет интерфейсы для подключения плагинов (например, машинного обучения).

– FileUpload

Сервис-обертка над Minio. Реализует функционал формирования ссылки для загрузки и выгрузки файлов.

– Minio

Сервис хранилище для файлов. Здесь хранятся, файлы, архивы, скрипты. Процесс загрузки/получения файлов в minio реализуется в 2 этапа.

- Получение ссылки через FileUpload для размещения или получения файла
- Загрузка/получение в minio файла напрямую через ссылку.

– PostgresDB

Используемая на проекте БД. Обслуживает Camunda и PlatromAPI

– RoleService

Сервис для ведения ролевой системы и определения доступности пользователю того или иного функционала системы. Сервис базируется на функциональность KeyCloak.

2.3 Ландшафт ведения разработки

Ландшафт разработки представлен тремя контурами. В дальнейшем могут быть добавлены новые контуры под различные сценарии.

1. DEV (контур разработки) предназначен для совместного тестирования ПО, демонстрации результатов перед релизом. Доступен всем участникам процесса разработки.
2. QA (Тестовый контур) предназначен для тестирования release, а также для демонстрации возможностей ПО в ходе тестирования. Представлен виртуальной машиной, в которой собран демонстрационный образец разрабатываемого ПО. На QA приложение переносится после формирования release.
3. PROD (Демонстрационный стенд) предназначен для показа инвесторам, заказчикам и остальным заинтересованным лицам. Представлен виртуальной машиной, в которой собран демонстрационный образец разрабатываемого ПО из ветки.

Варианты развёртывания ПО:

1. docker.
2. Установка на хост исполняемых файлов.

На первом этапе готовые образы, скрипты, оборудование должны быть в одной сети. Для теста и показа взаимодействия между разными сетями или ЦОД, контуры могут быть продублированы.

2.4 Описание технологического стека

Описание технологического стека части front, см.Таблица 1.

Таблица 1 – Технологический стек части front

Компонент\ Язык\ Фреймворк\ Библиотека	Лицензия	Ссылка
axios	MIT	https://www.npmjs.com/package/axios

core-js	MIT	https://www.npmjs.com/package/core-js
lodash.debounce	MIT	https://www.npmjs.com/package/lodash.debounce
lodash.isequal	MIT	https://www.npmjs.com/package/lodash.isequal
vue	MIT	https://www.npmjs.com/package/vue
vue-codemirror	MIT	https://www.npmjs.com/package/vue-codemirror
vue-router	MIT	https://www.npmjs.com/package/vue-router
vuelidate	MIT	https://www.npmjs.com/package/vuelidate
vuetify	MIT	https://www.npmjs.com/package/vuetify
vuex	MIT	https://www.npmjs.com/package/vuex

Описание технологического стека части back см. Таблица 2.

Таблица 2– Технологический стек части back

Компонент\ Язык\ Фреймворк\ Библиотека	**Лицензия**	**Ссылка**
fasthttprouter	BSD 3	github.com/buaazp/fasthttprouter
camunda-клиент-go	MIT	github.com/citilinkru/camunda-client-go
JsonConfigReader	MIT	github.com/DisposaBoy/JsonConfigReader
runtime	Apache 2.0	github.com/go-openapi/runtime
pg	MIT	github.com/go-pg
migrat	MIT	github.com/golang-migrate/migrat
grpc-middleware	Apache 2.0	github.com/grpc-ecosystem/go-grpc-middleware
grpc-prometheus	Apache 2.0	github.com/grpc-ecosystem/go-grpc-prometheus
grpc-gateway	BSD 3	github.com/grpc-ecosystem/grpc-gateway
sqlx	MIT	github.com/jmoiron/sqlx

easyjson	MIT	github.com/mailru/easyjson
minio-go	Apache 2.0	github.com/minio/minio-go
gocloak	Apache 2.0	github.com/Nerzal/gocloak
opentracing-go	Apache 2.0	github.com/opentracing/opentracing-go
errors	BSD 2	github.com/pkg/errors
client_golang	Apache 2.0	github.com/prometheus/client_golang
go.uuid	MIT	github.com/satori/go.uuid
logrus	MIT	github.com/sirupsen/logrus
spf13/viper	MIT	github.com/spf13/viper
testify	MIT	github.com/stretchr/testify
grpc-websocket-proxy	MIT	github.com/tmc/grpc-websocket-proxy
fasthttp	MIT	github.com/valyala/fasthttp
grpc	Apache 2.0	google.golang.org/grpc
protobuf	BSD-3	google.golang.org/protobuf

3 Подготовка к работе

3.1 Получение доступа к Системе

Доступы разработчику для разработки компонентов Системы выдает администратор.

Доступы соответствуют проектной роли разработчика и той части функционала, над которой он работает.

Доступ к Git является общим.

3.2 Требования к рабочим местам

Для обеспечения выполнения работ по разработке компонентов Системы необходимо наличие на рабочих компьютерах:

1. Git;
2. Редактор исходного кода;
3. Используемых средств разработки.

Требования к техническому обеспечению рабочих компьютерах разработчиков определяются используемыми средствами, минимальные требования, следующие:

1. Процессор 4 ядра, не ниже Core i5 7xxx;
2. ОЗУ – не менее 16 ГБ;
3. Диск – не менее 250 ГБ;
4. Монитор с минимальным разрешением экрана не менее 1920x1080 пикселей;
5. Пропускная способность сети – не менее 10 Мбит/с.

4 Описание процесса разработки и тестирования Системы

4.1 Общие требования

Должны выполняться следующие общие требования:

a) Система опционально должно развертываться при помощи инструментов непрерывной интеграции и развертывания;
b) Должны выполняться все типовые технические требования к Системе;
c) Система должно функционировать в веб-браузере Яндекс.Браузер версии 14 и выше.

4.2 Требования к оформлению программы

Программа разрабатывается в средах:

a) Visual Code;
b) Vim.

Настройки лежат в репозиториях проекта Bergen IPA.

Перечень репозиториев проекта Bergen IPA представлен ниже, см. Таблица

Таблица 3 – Список репозиториев

Репозиторий в Git	Ветки	Ссылки
api-gw	ros	https://gitlab.bergen.tech/ips/api-gw/-/tree/ros
base-lib	ros	https://gitlab.bergen.tech/ips/base-lib/-/tree/ros
camunda-wrapper	ros	https://gitlab.bergen.tech/ips/camunda-wrapper/-/tree/ros
docker-compose	ros	https://gitlab.bergen.tech/ips/docker-compose/-/tree/ros
file-upload	ros	https://gitlab.bergen.tech/ips/file-upload/-/tree/ros

ips-ui	ros	https://gitlab.bergen.tech/ips/ips-ui/-/tree/ros
keycloack-role-wrapper	ros	https://gitlab.bergen.tech/ips/keycloack-role-wrapper/-/tree/ros
ml-runner	ros	https://gitlab.bergen.tech/ips/ml-runner/-/tree/ros
model-service	ros	https://gitlab.bergen.tech/ips/model-service/-/tree/ros
platforma-api	ros	https://gitlab.bergen.tech/ips/platforma-api/-/tree/ros
reference-book	ros	https://gitlab.bergen.tech/ips/reference-book/-/tree/ros
reporting	ros	https://gitlab.bergen.tech/ips/reporting/-/tree/ros

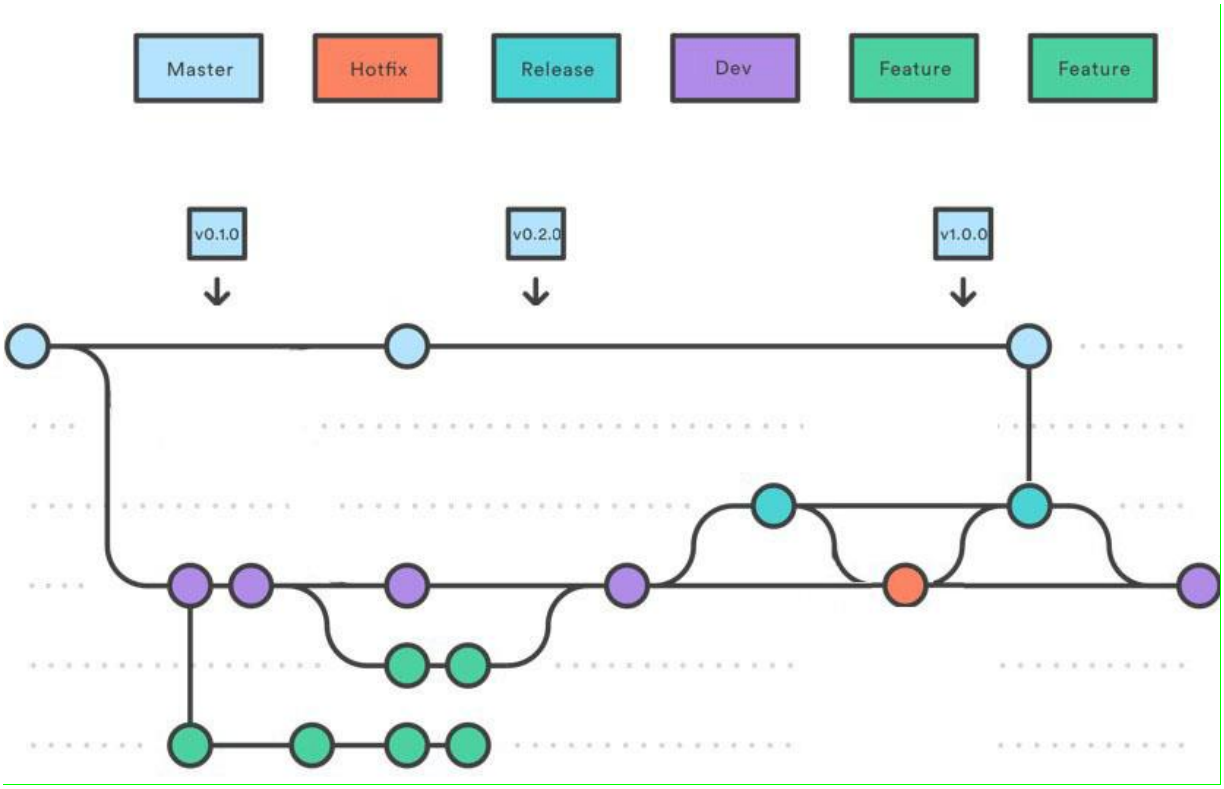
4.3 Регламент ведения разработки

1. Git-репозитории разделены по типам приложений (фронтенд, бэкенд, мидл, воркеры)
2. По умолчанию перед сборкой запускается линтер (eslinter).
3. Разработчик отвечает за отсутствие merge-конфликтов при слиянии в ветку dev.
4. Каждый метод должен быть задокументирован в системе документации.
5. Контроль технической документации и код-ревью перед переносом в продуктив осуществляет тимлид проекта.
6. Именование разработчиков ведётся в Active Directory, так же и авторизация в git идёт через AD.
7. MR (merge request) в DEV ветку создает разработчик, выполняющий задачу. Само выполнение MR должен делать не тот разработчик, который создал MR. 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 Merge.

4.4 Работа с Git

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

- master: в данной ветке хранится продакшн-версия проекта, при этом изменения в этой ветке не производятся напрямую, а вливаются из develop ветки.
- 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.
- release/release-x.x.x: как только dev ветка набрала в себя достаточно изменений и релиз менеджер считает, что пора выпустить новую версию, необходимо ответвится 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.
- 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 develop через merge request. При выполнении большой задачи в течении долгого времени обязательно каждые 1-2 дня мерджить к себе основную ветку разработки, чтобы избежать накопления большой разницы в коде и тяжело решаемых конфликтов изменений.
- 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.



Разработчик может выполнять commit в ветку ros - * самостоятельно.

Ветка ros содержит последнюю рабочую версию разрабатываемого ПО.

4.5 Структура репозиториев Git

Типовая структура репозитория:

app — исходный код приложения. В зависимости от типа приложения имеет специфичную структуру для данного приложения;

.envvars — список переменных окружения;

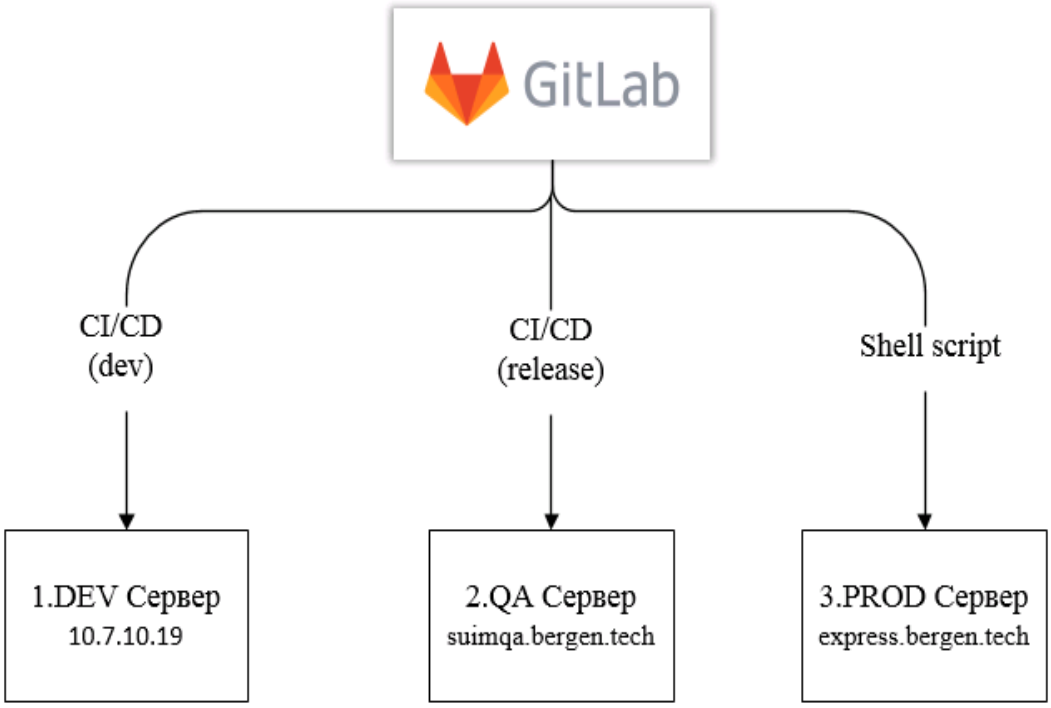
Dockerfile — файл описания сборки docker image для приложения.

.ci-env — список переменных окружения необходимый для CI/CD gitlab;

README.md — описание приложения, расположенного в репозитории, техническая информация для разработчика, описания и краткие инструкции;

gitlab-ci.yml — файл pipline этапов для CI/CD gitlab описывающий проверку, сборку, тестирование и релиз приложения.

4.6 Порядок выпуска релизов



1.DEV сервер

На этот сервер разворачивается приложение средствами CI/CD gitlab из ветки dev. Сервер находится в локальной сети Берген с возможностью доступа из вне с использованием VPN. Переносы на данный сервер выполняются в любой момент по желанию разработчика с уведомлением в чат скайп в группу разработчиков.

2. QA сервер

Обновляется по согласованию в чате разработчиков и тестировщиков. По согласованию с группой тестировщиков допускаются переносы в любое время.

Обновления тестового сервера соответствуют плану релизов.

3. PROD сервер

Обновляется после выпуска релиза по согласованию с руководителем мастер группы. Перенос осуществляется запуском sh скрипта на сервере и, если необходимо с исправлением конфигурации, в соответствующем репозитории. Обновление производит только devops специалист/администратор.

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.

Общий порядок работы команды над задачей

Пример:

a) В RedMine создается задача с номером № 100000 и названием «Доработка backend части авторизации», назначается на разработчика;

b) Когда разработчик приступает к задаче, то должен изменить статус задачи на «В Работе» и создать бранч от **dev** ветки с названием *"feature/<name>*-100000*"*, в котором он будет вести свою работу. В данном примере <name> задается семантически по смыслу задачи;

c) 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 (назначить отдельного merge request approval нельзя в бесплатной версии gitlab);

d) Текст коммита должен соответствовать следующему формату: *«#NNNNN <Описание изменений>»,* где NNNNN – номер задачи в RedMine;

e) В тексте коммита желательно придерживаться обезличенных формулировок. То есть «исправлена проблема, связанная с ….», «добавлен функционал позволяющий …..»;

f) 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. То есть статус задачи «В тесте» означает, что разработка закончена, локально протестирована, код смержен в dev ветку и развернут на QA сервере;

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

h) После формирования релиза администратор переводит задачи из статуса «В релиз» в статус «Решена».

i) Статус «Решена» – код находится в release ветке, успешно был протестирован на **QA** сервере и готов к развороту на конечном **PROD** сервер.

Описание API

Разрабатываемый API должен быть описан в OpenAPI (бывший swagger).

Асинхронные взаимодействия должны быть описаны в OpenAPI (например, с помощью callback).

Описание отдельных сообщений при взаимодействии делается с помощью JSON схемы.

Корректность формата JSON можно проверить тут https://jsonform.github.io/jsonform/playground/index.html.

Протокол взаимодействия (последовательность обмена сообщениями) необходимо дополнительно описать в текстовом виде либо с помощью sequence диаграммы.

4.7 Нумерация релизов

Каждому релизу должен быть присвоен номер версии. Этот номер, или тэг, будет являться уникальным для каждого отдельного релиза, храниться в истории проекта и станет понятным для любого разработчика/менеджера проекта. Общепринятый формат нумерации релиза X.Y.Z. В случае выпуска релиза для каждого компонента системы делается новая версия даже если в компоненте не было изменений. Соответственно при установке все компоненты должны быть одной версии. Семантическое версионирование не используется по вышеописанным причинам.

Учитывая номер версии: мажорная, минорная, патч, следует увеличивать:

a) Мажорную версию, когда сделаны обратно несовместимые изменения API;

b) Минорную версию, когда добавлена новая функциональность, не нарушающая обратную совместимости;

c) Патч-версию, когда сделаны обратно совместимые исправления.

Дополнительные обозначения для предрелизных и билд-метаданных возможны как дополнения к мажорному, минорному, патч-формату.

4.8 Интеграция с RedMine

Заведение задач на проекте ведётся в системе управления проектами RedMine. При этом для каждого проекта проводится интеграция системы с Git-репозиторием. Поэтому при выполнении коммита желательно придерживаться следующего шаблона: Fix some bug issue #9940 @1h30

В каждом коммите необходимо указывать номер задачи, в рамках которой выполнялась разработка (#9940) и время, затраченное на разработку (@1h30 – в данном случае 1 час 30 минут). Такой коммит будет отображаться в RedMine на вкладке «Хранилище», см. Рисунок 2.

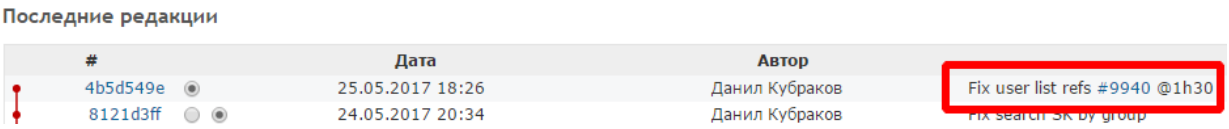
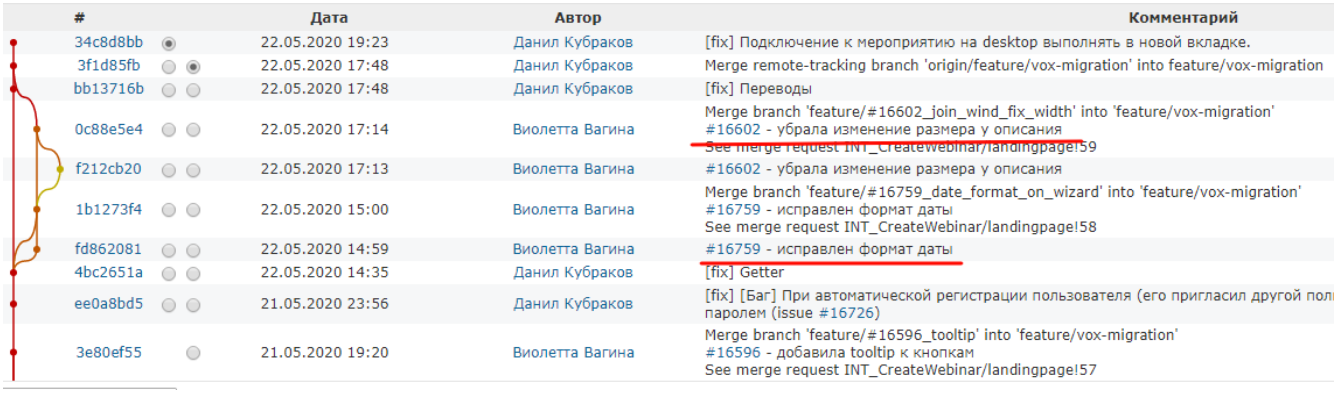


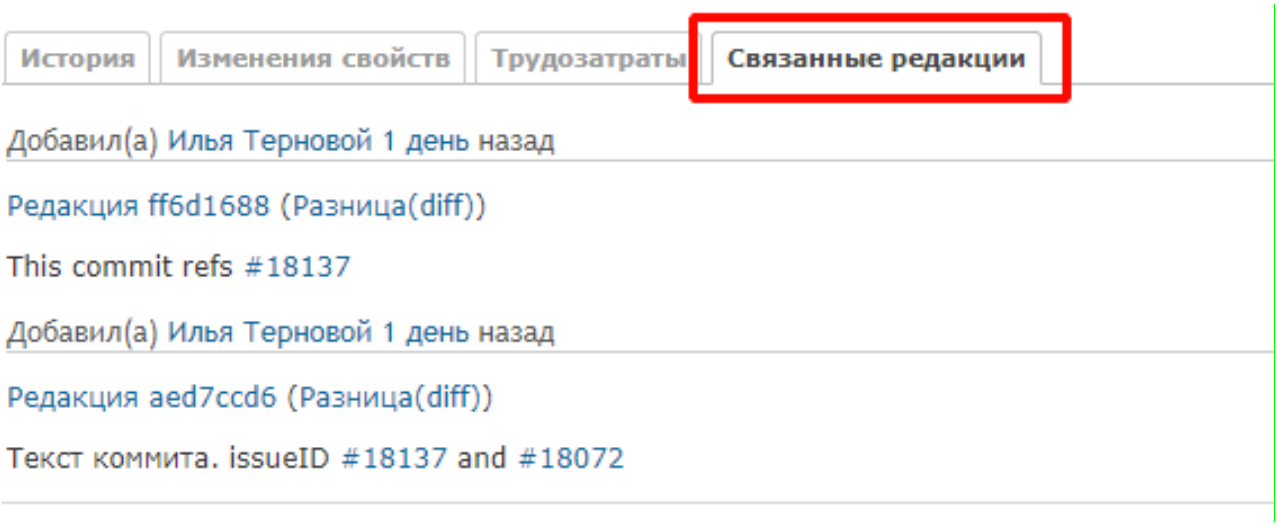
Рисунок 2 – Коммиты в RedMine

Заведение задач на проекте ведётся в системе управления проектами Redmine. При этом для каждого проекта проводится интеграция системы с Git-репозиторием. Поэтому при выполнении коммита необходимо придерживаться следующего шаблона: «refs #9940 Исправлен баг с невозможностью добавления контактов при редактировании конференции», то есть в каждом коммите необходимо указывать ключевое слово refs и номер задачи, в рамках которой выполнялась разработка (#9940). Такой коммит будет отображаться в Redmine на вкладке «Хранилище» следующим образом:



#	Дата	Автор	Комментарий
34c8d8bb	22.05.2020 19:23	Данил Кубраков	[fix] Подключение к мероприятию на desktop выполнять в новой вкладке.
3f1d85fb	22.05.2020 17:48	Данил Кубраков	Merge remote-tracking branch 'origin/feature/vox-migration' into feature/vox-migration
bb13716b	22.05.2020 17:48	Данил Кубраков	[fix] Переводы
0c88e5e4	22.05.2020 17:14	Виолетта Вагина	Merge branch 'feature/#16602_join_wind_fix_width' into 'feature/vox-migration' #16602 - убрала изменение размера у описания See merge request INT_CreateWebinar/landingpage!59
f212cb20	22.05.2020 17:13	Виолетта Вагина	#16602 - убрала изменение размера у описания
1b1273f4	22.05.2020 15:00	Виолетта Вагина	Merge branch 'feature/#16759_date_format_on_wizard' into 'feature/vox-migration' #16759 - исправлен формат даты See merge request INT_CreateWebinar/landingpage!58
fd862081	22.05.2020 14:59	Виолетта Вагина	#16759 - исправлен формат даты
4bc2651a	22.05.2020 14:35	Данил Кубраков	[fix] Getter
ee0a8bd5	21.05.2020 23:56	Данил Кубраков	[fix] [Баг] При автоматической регистрации пользователя (его пригласил другой пол паролем (issue #16726)
3e80ef55	21.05.2020 19:20	Виолетта Вагина	Merge branch 'feature/#16596_tooltip' into 'feature/vox-migration' #16596 - добавила tooltip к кнопкам See merge request INT_CreateWebinar/landingpage!57

И на вкладке самой задачи:



5 Реализация требований по обеспечению информационной безопасности Системы

Информационная безопасность предполагает обеспечение защиты данных от хищений или изменений как случайного, так и умышленного характера.

Система можно условно разделить на несколько составляющих:

a) Frontend;

b) Backend.

Информационная безопасность подразумевает доступ к СУБД исключительно из Backend модулей. А работа с Системам осуществляется

исключительно с помощью Frontend. Ни к Backend, ни к СУБД пользователь напрямую доступа не имеет. Для работы с СУБД в Backend реализован ряд сервисов, к которым обращается Frontend.

В зависимости от роли пользователя, ему предоставляются определенные права и полномочия.

6 Аварийные ситуации

6.1 Действия в случае обнаружения ошибок в Приложении

В случае возникновения ошибок в работе компонентов Системы пользователям необходимо зафиксировать проблему подробно в виде письма на адрес службы поддержки. Служба поддержки на основании писем с заголовком «Проблема Bergen Intelligent Process Automation» формирует задачи RM. Параметры задачи (рекомендуется прикладывать скриншоты ошибки):

a) Назначать руководителю разработки (АРП);

b) Статус задачи «Открыто»;

c) Приоритет проставляется в зависимости от критичности ошибки;

d) Описание ошибки обязательно должно включать в себя алгоритм получения ошибки.

В дальнейшем задача будет назначена на разработчика.

6.2 Действия по восстановлению программ и/или данных при отказе носителей информации или обнаружении ошибок в данных

Выполнение процедуры восстановления данных может быть необходимо по причине системного сбоя, который привел к полной или частичной потере данных, или по причине неверных действий разработчика, которые привели к изменению/потере данных. Для восстановления программ и/или данных необходимо зафиксировать запрос в виде задачи RM на АДМ. Приоритет задачи «Наивысший».

6.3 Действия при обнаружении несанкционированного вмешательства в данные

При обнаружении несанкционированного вмешательства в данные необходимо зафиксировать запрос в виде задачи RM на АДМ с приоритетом задачи «Наивысший».

7. Обращение в службу технической поддержки

При возникновении проблем во время работы с системой, обращайтесь в службу технической поддержки.

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

поддержки по адресу support@bergen.tech или обращаться по телефону 8 (495) 664-37-83.

Перечень ссылочных документов

Руководство пользователя, 2021
Руководство администратора, 2021

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	12.07.2021	Суркина Лилия	Аналитик	
0.2.	29.07.2021	Жарков Анатолий	Руководитель разработки	
1.0	29.07.2021	Вострухина Елизавета	Аналитик	
1.1	30.07.2021	Суркина Лилия	Аналитик	
1.2	30.07.2021	Вострухина Елизавета	Аналитик	