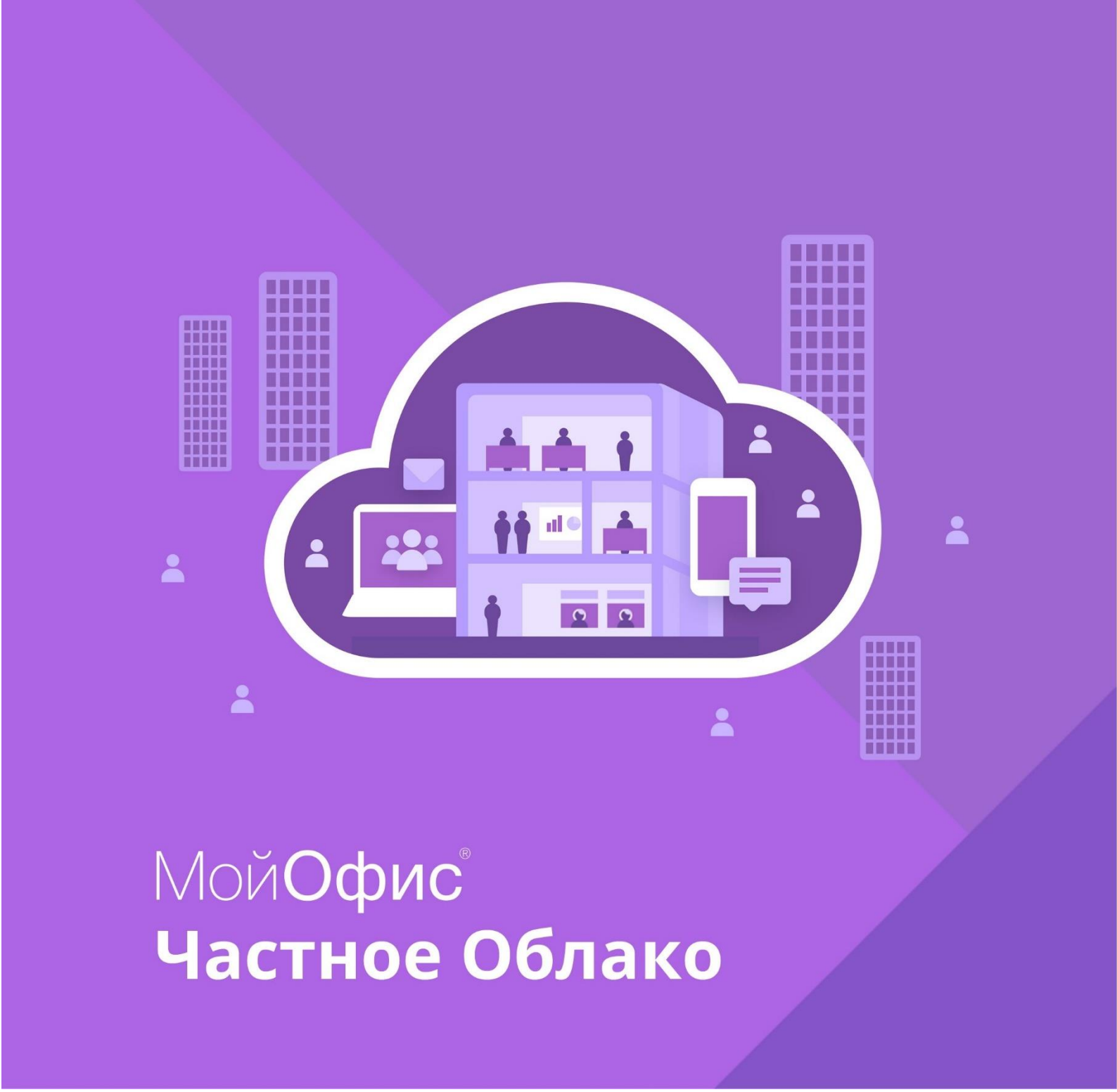

МойОфис®
Частное Облако

Сервисно-ресурсная модель

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«МОЙОФИС ЧАСТНОЕ ОБЛАКО»

СЕРВИСНО-РЕСУРСНАЯ МОДЕЛЬ

2020.02.R2

На 47 листах

Москва
2020

Все упомянутые в этом документе названия продуктов, логотипы, торговые марки и товарные знаки принадлежат их владельцам.

Товарные знаки «МойОфис» и «MyOffice» принадлежат ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ».

Ни при каких обстоятельствах нельзя истолковывать любое содержимое настоящего документа как прямое или косвенное предоставление лицензии или права на использование товарных знаков, логотипов или знаков обслуживания, приведенных в нем. Любое несанкционированное использование этих товарных знаков, логотипов или знаков обслуживания без письменного разрешения их правообладателя строго запрещено.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, ТЕРМИНОВ И ОПРЕДЕЛЕНИЙ

В настоящем документе применяют следующие сокращения с соответствующими расшифровками (см. Таблица 1):

Таблица 1 – Сокращения и расшифровки

Сокращение, термин	Расшифровка и определение
ОС	Операционная система
CPM	Сервисно-ресурсная модель. Логическая модель сервиса, описывающая состав и взаимосвязи компонентов
API	Application programming interface (анг.), программный интерфейс приложения — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой
CDN	Content Delivery Network или Content Distribution Network (анг.), сеть доставки (и дистрибуции) содержимого — распределённая сетевая инфраструктура, позволяющая оптимизировать доставку и дистрибуцию содержимого конечным пользователям в сети Интернет
CO	Cloud Office (анг.), Облачный Офис, общее название продукта
CVM	Conversion Manager (анг.), Сервис управления конвертированием файлов
CU	Conversion Unit(анг.), экземпляр процесса конвертора различных форматов файлов
DCM	Document Collaboration Manager (анг.), Сервис управления редактированием документов
DU	Document Unit (анг.), экземпляр процесса редактирования и коллаборации документов
PGS	Pythagoras Storage (анг.). Компонент сервиса МойОфис Частное Облако, представляет собой объектное хранилище
FM	File Manager (анг.). Сервис файлового менеджера
HA	High Availability (анг.), высокая доступность — характеристика технической системы, разработанной во избежание невыполненного обслуживания путём уменьшения или управления сбоями и минимизацией времени плановых простоев
JOD	Java OpenDocument Converter (анг.). Сервис конвертирования документов стандарта OpenDocument
NM	Notification Manager (анг.). Сервис управления оповещениями
PSN	Postal Solution (анг.). Компонент сервиса МойОфис Частное Облако, представляет собой почтовый сервис, сервис календарей и контактов.
SSO	Single Sign-On (анг.), технология единого входа — технология, при использовании которой пользователь переходит из одной системы в другую, не связанную с первой системой, без повторной аутентификации.
WFE	Web Frontend (анг.). Общее название web приложений МойОфис Частное Облако

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Введение

Сервисно-ресурсная модель (СРМ) – это логическая модель сервиса, описывающая состав и взаимосвязи компонентов (ресурсов), которые совместно обеспечивают предоставление сервиса. Как правило, СРМ имеет графическое представление в виде иерархического графа, узлами которого являются компоненты, а ребрами – связи между ними. СРМ может быть использована для решения широкого круга задач, однако в первую очередь предназначена для мониторинга параметров качества сервиса, который может выполняться в рамках процесса управления доступностью. В данном документе графическое представление СРМ приведено в разделе Диаграммы, а методики проверки значений параметров сервиса - в соответствующих разделах.

1.2 Градации и определение степени влияния Сервисов на Подсистемы

В данном документе используются несколько уровней влияния отдельного Сервиса на Подсистему и на Систему в целом. Уровни определены ниже, в Таблица 2.

Таблица 2. Уровни влияния сервисов на Подсистему

Уровень влияния	Описание
Critical	Отказ Сервиса приводит к полной неработоспособности Подсистемы
Major	Отказ Сервиса не приводит к неработоспособности Подсистемы/Отказ Сервиса приводит к отсутствию критичной функциональности при общей доступности Сервиса
Minor	Отказ Сервиса приводит к неработоспособности редко используемой функциональности, либо к падению производительности
Warning	Отказ Сервиса не приводит потере функциональности (например, отказ одного из узлов кластера)

2 ДИАГРАММЫ

2.1 Сервис МойОфис Частное Облако

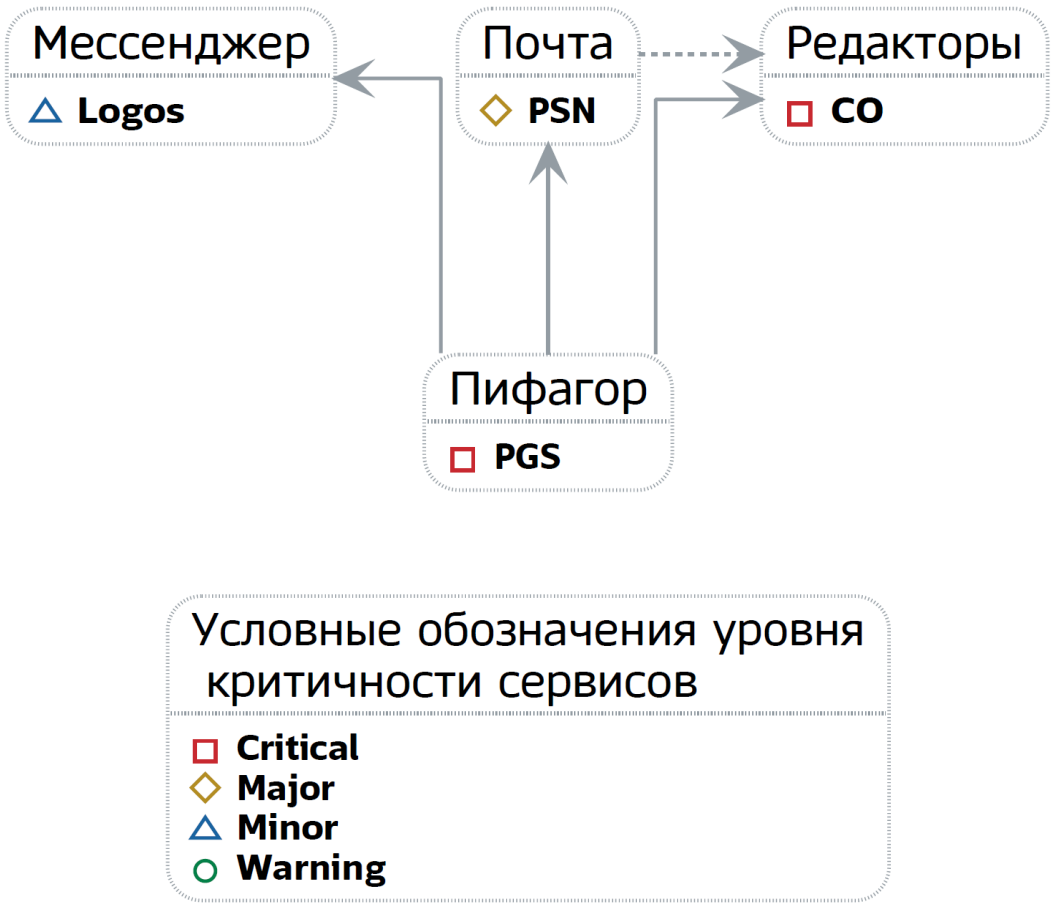

Рисунок 1 – Сервисно-ресурсная модель МойОфис Частное Облако

2.2 Сервисно-ресурсная модель Редакторы (CO)

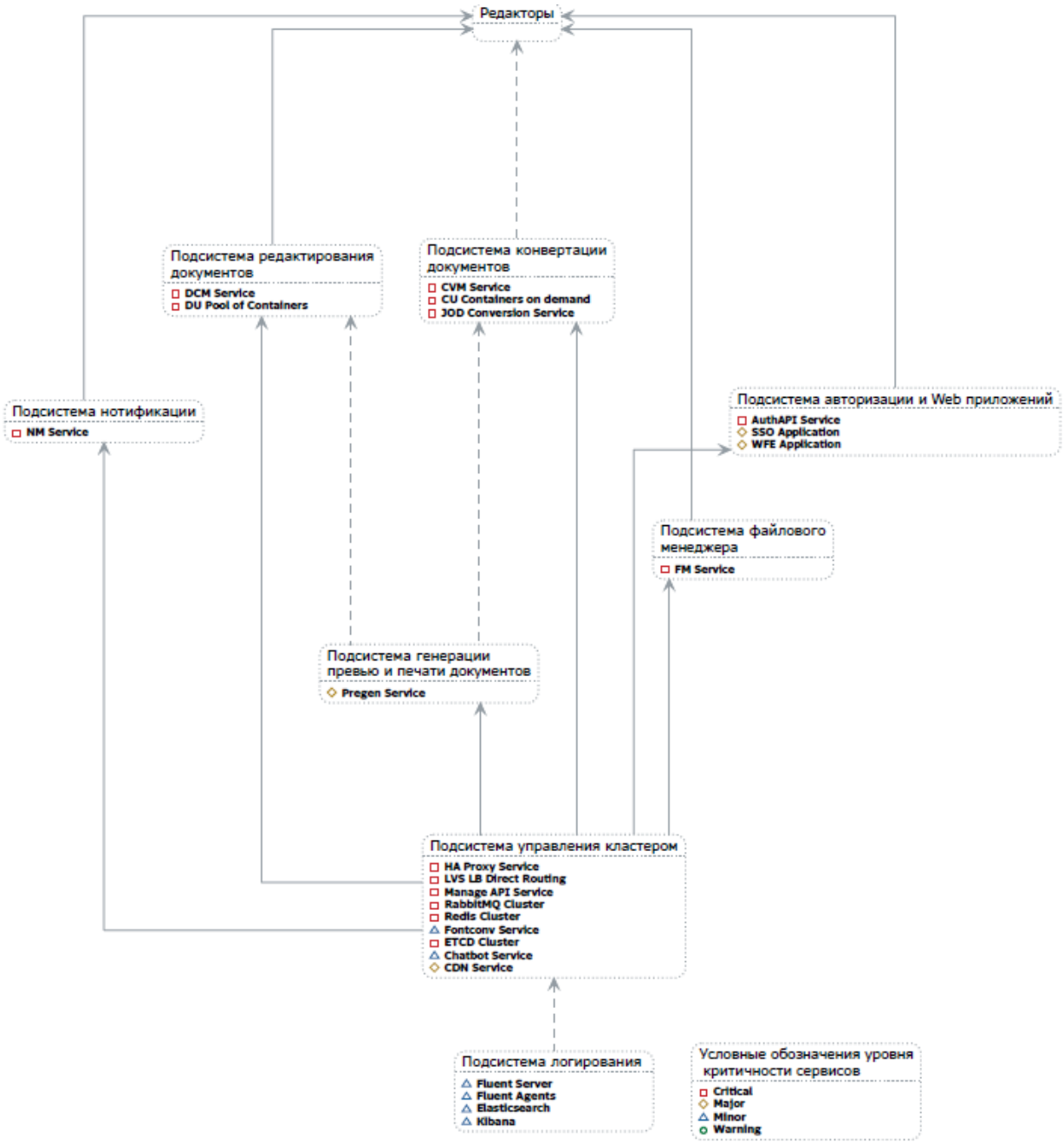

Рисунок 2 – Сервисно-ресурсная модель Редакторы(CO)

2.3 Сервисно-ресурсная модель Почта (PSN)

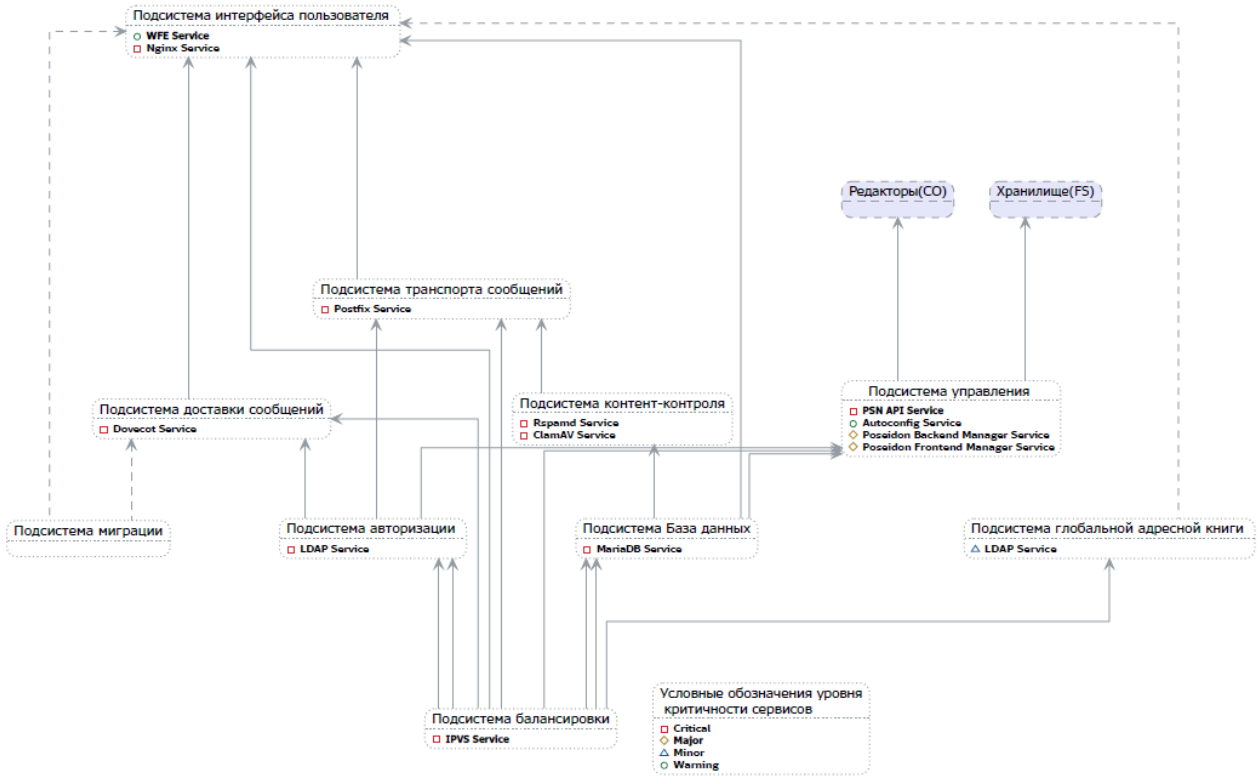

Рисунок 3 – Сервисно-ресурсная модель Почта (PSN)

2.4 Сервисно-ресурсная модель Мессенджер (Logos)

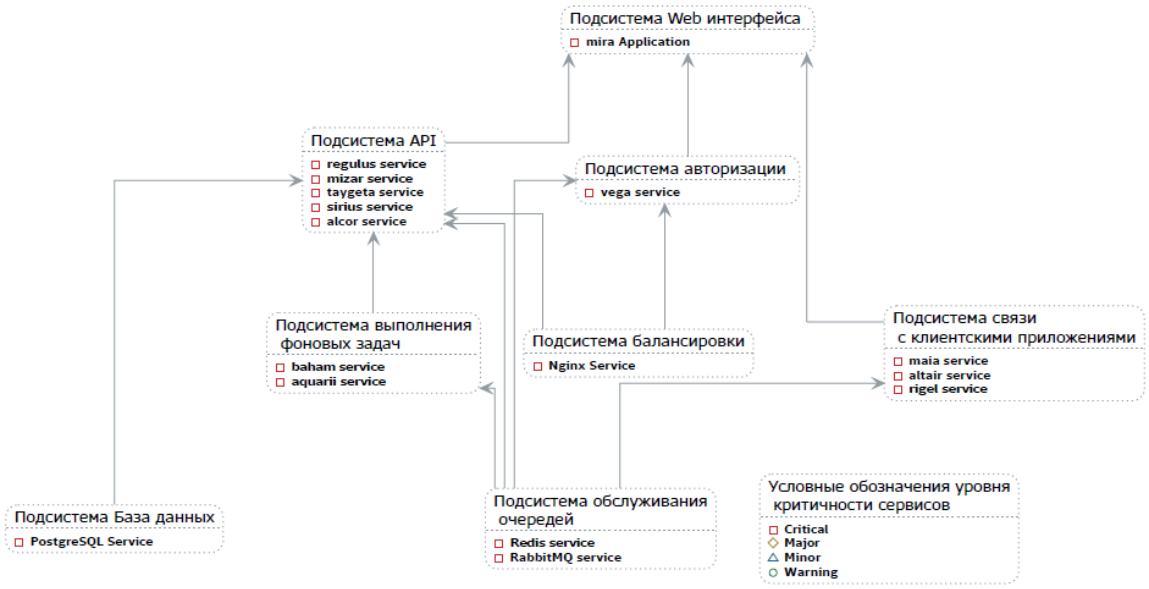

Рисунок 4 – Сервисно-ресурсная модель Мессенджер(Logos)

2.5 Сервисно-ресурсная модель Пифагор(PGS)

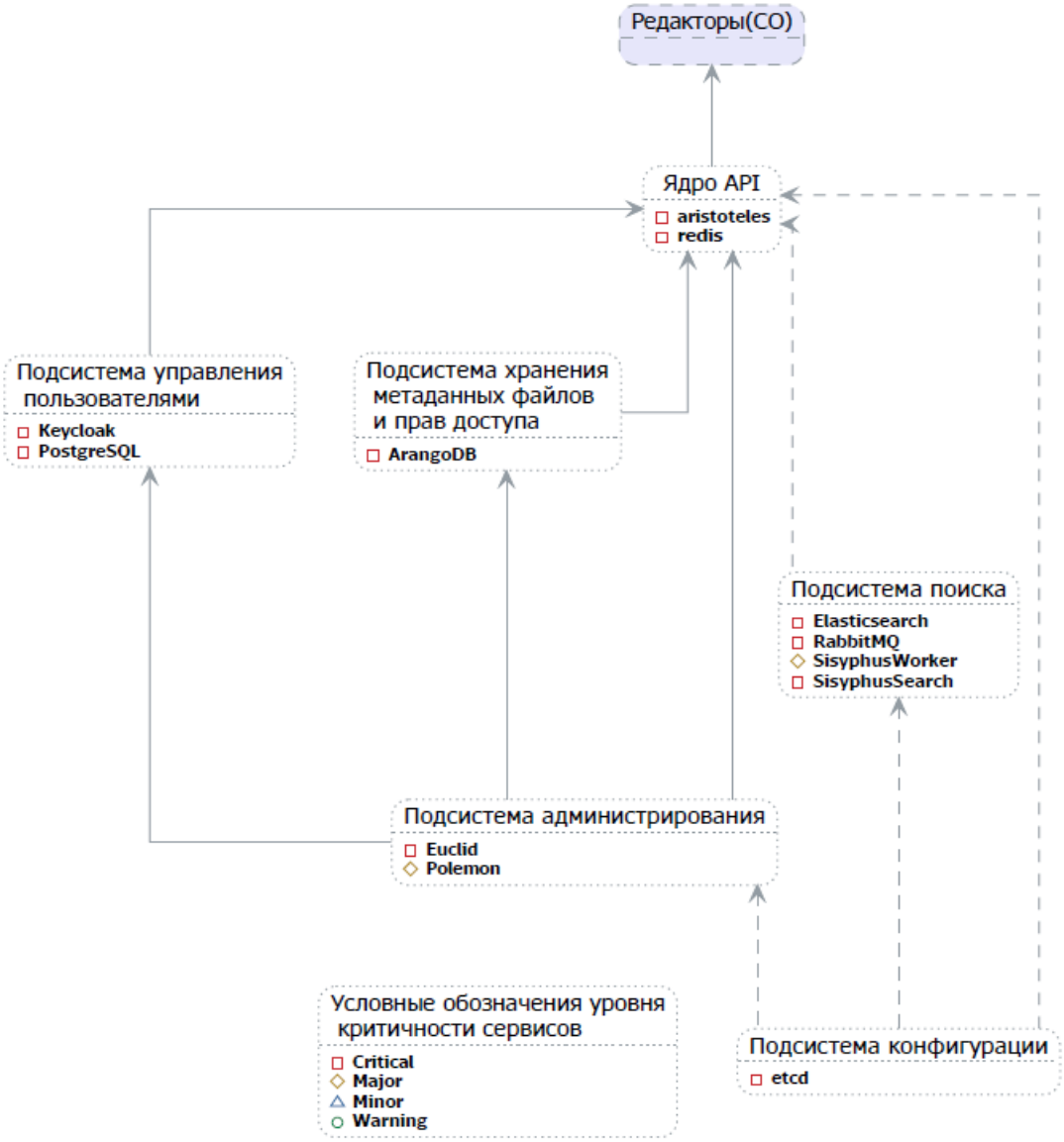

Рисунок 5 – Сервисно-ресурсная модель Пифагор (PGS)

3 РЕДАКТОРЫ (CO)

В описании методов контроля используются переменные вида **$PRIVATE_IPV4, $CVM_PORT** или **$DCM_CHECK_PATH**, которые определяют параметры конкретного окружения. Значения этих переменных определены в файлах, находящимся на каждом хосте. Переменная **$PRIVATE_IPV4** определена в **/etc/environment**, остальные переменные - в файле **/opt/co/systemd/systemd-env** Расположение файла может быть иным, отличным от **/opt/co**, если при развертывании было указано иное значение переменной **CO_DIR**.

Контроль функционирования сервисов выполняется запросами из командной строки, вызовами утилиты **curl**. Весь список переменных, используемых в командах проверки, можно определить командами:

```
set -a
source /opt/co/systemd/systemd-env
set +a
```

и в дальнейшем применять команды согласно настоящей методике, т.е.:

```
curl -s $CVM_SCHEME://$CVM_HOST:$CVM_PORT/$CVM_CHECK_PATH | jq '.status'
```

3.1 Подсистема авторизации и Web приложений

Таблица 3. Параметры Подсистемы авторизации и Web приложений

Контролируемый параметр	Корректное значение	Критичность
"Auth API"	"OK"	Critical
"TLS GOST Proxy"	"running"	Critical
"SSO Application"	"running"	Major
"WFE Application"	"running"	Major
"WTE Application"	"running"	Major

3.1.1 Методика контроля AuthAPI

Проверка статуса работы подсистемы AuthAPI осуществляется командной строке:

```
curl -s
http://$PRIVATE_IPV4:$OPENRESTY_LB_CORE_AUTH_MNG_PORT/api/manage/core/sta
tus | jq '.all'
```

Пример ответа:

```
"OK"
```

3.1.2 Методика контроля TLS GOST Proxy

ВНИМАНИЕ	Данная проверка распространяется только на версии Частного Облака, содержащие криптографические преобразования согласно ГОСТ РФ

Для проверки статуса TLS GOST требуется убедиться в том, что контейнер с приложением запущен, для этого в терминале от пользователя **root** выполнить команду:

```
curl -s --unix-socket /var/run/docker.sock http://v1.38/containers/json |
jq '.[] | select(.Names == ["/nginx-gost"]) | .State'
```

Пример ответа:

```
"running"
```

3.1.3 Методика контроля параметров SSO Application

Для проверки статуса SSO Application на всех хостах с ролью **lb-core-auth** убедиться в том, что контейнер с приложением запущен, для этого в терминале от пользователя **root** выполнить команду:

```
curl -s --unix-socket /var/run/docker.sock http:/v1.38/containers/json |
jq '.[] | select(.Names == ["/web-sso"]) | .State'
```

Пример ответа:

```
"running"
```

3.1.4 Методика контроля параметров WFE Application

Для проверки статуса WFE Application на всех хостах с ролью **lb-core-auth** убедиться в том, что контейнер с приложением запущен, для этого в терминале от пользователя **root** выполнить команду:

```
curl -s --unix-socket /var/run/docker.sock http:/v1.38/containers/json | jq '.[] | select(.Names == ["/web-wfe"]) | .State'
```

Пример ответа:

```
"running"
```

3.1.5 Методика контроля параметров WTE Application

Для проверки статуса WTE Application на всех хостах с ролью **lb-core-auth** убедиться в том, что контейнер с приложением запущен, для этого в терминале от пользователя **root** выполнить команду:

```
curl -s --unix-socket /var/run/docker.sock http:/v1.38/containers/json | jq '.[] | select(.Names == ["/web-wte"]) | .State'
```

Пример ответа:

```
"running"
```

3.2 Подсистема файлового менеджера

Таблица 4. Параметры Подсистемы файлового менеджера

Контролируемый параметр	Корректное значение	Критичность
"FM Service"	"UP"	Critical

3.2.1 Методика контроля FM Service

```
curl -s $FM_SCHEME://$FM_HOST:$FM_PORT/$FM_CHECK_PATH | jq '.status'
```

Пример ответа:

```
"UP"
```

3.3 Подсистема конвертации документов

Таблица 5. Параметры Подсистемы конвертации документов

Контролируемый параметр	Корректное значение	Критичность
"CVM Service"	"UP"	Critical
"CU Containers on demand"	"true"	Critical
"JOD Conversion Service"	"UP"	Critical

3.3.1 Методика контроля CVM Service

Проверка статуса работы подсистемы CVM Service осуществляется в командной строке:

```
curl -s $CVM_SCHEME://$CVM_HOST:$CVM_PORT/$CVM_CHECK_PATH | jq '.status'
```

Пример ответа:

```
"UP"
```

3.3.2 Методика контроля параметров CU Containers on demand

Проверка статуса работы подсистемы CU Containers on demand осуществляется в командной строке:

```
curl -s $CVM_SCHEME://$CVM_HOST:$CVM_PORT/$CVM_CHECK_PATH | jq
'.components.CVM.details.npsPool > 0'
```

Пример ответа:

```
"true"
```

3.3.3 Методика контроля параметров JOD Conversion Service

```
curl -s $JOD_SCHEME://$PRIVATE_IPV4:$JOD_PORT/$JOD_CHECK_PATH | jq
'.status'
```

Пример ответа:

```
"UP"
```

3.4 Подсистема генерации превью и печати документов

Таблица 6. Параметры Подсистемы генерации превью и печати документов

Контролируемый параметр	Корректное значение	Критичность
"Pregen Service"	"OK"	Major

3.4.1 Методика контроля Pregen Service

Команда контроля **Pregen**:

```
curl -sX HEAD -I
$PREGEN_SCHEME://$PREGEN_HOST:$PREGEN_PORT$PREGEN_CHECK_PATH | grep HTTP
| awk '{print $3 }'
```

Пример ответа:

```
"OK"
```

3.5 Подсистема редактирования документов

Таблица 7. Параметры Подсистемы редактирования документов

Контролируемый параметр	Корректное значение	Критичность
"DCM Service"	"UP"	Critical
"DU Pool of Containers"	"ok"	Critical

3.5.1 Методика контроля DCM Service

```
curl -s $DCM_SCHEME://$DCM_HOST:$DCM_PORT/$DCM_CHECK_PATH | jq '.status'
```

Пример ответа:

```
"UP"
```

3.5.2 Методика контроля параметров DU Pool of Containers

Количество запущенных контейнеров DU должно находиться в пределах, определенных значениями переменных **docker.duPoolSize** и **docker.duPoolSizeMax**. Значения

переменных задаются в файле **/opt/co/etcd-config/dcm.properties**. Количество запущенных контейнеров определяется командой:

```
curl -s --unix-socket /var/run/docker.sock -X GET
http:/v1.38/containers/json | jq '[.[] | select(.Names | contains(
["/DU"]))] | length'
```

Полученное значение не должно быть меньше значения **docker.duPoolSize**

3.6 Подсистема нотификации

Таблица 8. Параметры Подсистемы нотификации

Контролируемый параметр	Корректное значение	Критичность
"NM Service"	"UP"	Critical

3.6.1 Методика контроля параметров NM Service

```
curl -s $NM_SCHEME://$PRIVATE_IPV4:$NM_PORT/$NM_CHECK_PATH | jq '.status'
```

Пример ответа:

```
"UP"
```

3.7 Подсистема управления кластером

Таблица 9. Параметры Подсистемы управления кластером

Контролируемый параметр	Корректное значение	Критичность
"Redis Cluster"	"PONG"	Critical
"RabbitMQ Cluster"	"ok"	Critical
"ETCD Cluster"	"true"	Critical
"HA Proxy Service"	"200 OK Service ready"	Critical
"LVS LB Direct Routing"	"true"	Critical
"Manage API Service"	"OK"	Critical
"Fontconv Service"	"OK"	Minor

"CDN Service"	"running"	Major
"Chatbot Service"	"OK"	Minor

3.7.1 Методика контроля Redis Cluster

Для проверки сервиса Redis Cluster предварительно на хост, с которого будет производится проверка, предварительно требуется установить утилиту **redis-cli**.

Для rpm-based дистрибутивов это можно сделатькомандой:

```
yum install redis
```

Для deb-based дистрибутивов это можно сделать командой:

```
sudo apt install redis-tools
```

После чего выполнить проверку корректности работы сервиса Redis Cluster

```
redis-cli -h $PRIVATE_IPV4 -p $REDIS_SENTINEL_PORT ping
```

Пример ответа:

```
"PONG"
```

3.7.2 Методика контроля RabbitMQ Cluster

```
curl -s
$RABBITMQ_SCHEME://$RABBITMQ_USERNAME:$RABBITMQ_PASSWORD@$RABBITMQ_NODE:$
RABBITMQ_MNG_PORT$RABBITMQ_CHECK_PATH | jq '.status'
```

Пример ответа:

```
"OK"
```

3.7.3 Методика контроля ETCD Cluster

```
curl -ks
https://$ETCD_BROWSER_USERNAME:$ETCD_BROWSER_PASSWORD@$PRIVATE_IPV4:$ETCD
_CLIENT_PORT/health | jq '.health'
```

Пример ответа:

```
"true"
```

3.7.4 Методика контроля HA Proxy Service

```
curl -s
$HAPROXY_SCHEME://$HAPROXY_USERNAME:$HAPROXY_PASSWORD@$PRIVATE_IPV4:$HAPR
OXY_PORT_STATS_HTTP$HAPROXY_CHECK_PATH |   sed -e 's/<[^>]*>//g'
```

Пример ответа:

```
200 OK
Service ready.
```

3.7.5 Методика контроля параметров LVS LB Direct Routing

Методика проверки LVS LB Direct Routing описана в разделе **Подсистема балансировки**

3.7.6 Методика контроля Manage API Service

На хостах с ролью **lb_core_auth** в консоли выполнить команду:

```
curl -s
http://$PRIVATE_IPV4:$OPENRESTY_LB_CORE_AUTH_MNG_PORT/api/manage/config/v
ersion
```

Ответ сервера с кодом 200 OK может считаться признаком штатной работы сервиса.

3.7.7 Методика контроля параметров Fontconv Service

```
curl -sX HEAD -I
$FONTCONV_SCHEME://$FONTCONV_HOST:$FONTCONV_PORT$FONTCONV_CHECK_PATH |
grep HTTP | awk '{print $3 }'
```

Пример отображения результата выполнения команды:

```
OK
```

3.7.8 Методика контроля параметров CDN Service

На хостах с ролями **core_fm , lb_core_auth, pregen** в консоли выполнить команду

```
systemctl is-active lsyncd
```

Корректным ответом будет **active**

3.7.9 Методика контроля параметров Chatbot Service

```
curl -sX HEAD -I
$CHATBOT_SCHEME://$CHATBOT_HOST:$CHATBOT_PORT$CHATBOT_CHECK_PATH | grep
HTTP | awk '{print $3 }'
```

Пример ответа:

```
OK
```

3.8 Подсистема логирования

Таблица 10. Параметры Подсистемы логирования

Контролируемый параметр	Корректное значение	Критичность
"Fluent Server"	"OK"	Minor
"Fluent Agents"	"OK"	Minor
"Elasticsearch"	green	Minor
"Kibana"	"OK"	Minor

3.8.1 Методика контроля параметров Fluent Server

Проверка проводится в командной строке на хостах с ролью **log**

```
docker exec -t fluentd-server curl -svo /dev/null
http://127.0.0.1:$FLUENTD_SERVER_PORT_MONITOR/api/plugins.json | grep '<
HTTP' | awk '{print $4 }'
```

Пример корректного ответа:

```
OK
```

3.8.2 Методика контроля параметров Fluent Agents

Проверка проводится в командной строке на всех хостах

```
docker exec -t fluentd-agent curl -svo /dev/null
http://127.0.0.1:$FLUENTD_AGENT_PORT_MONITOR/api/plugins.json | grep '<
HTTP' | awk '{print $4 }'
```

Пример корректного ответа:

```
OK
```

3.8.3 Методика контроля параметров Elasticsearch

Проверка проводится в командной строке на хостах с ролью **log**

```
docker exec -t fluentd-server curl -s "localhost:9200/_cluster/health" |
jq '.status'
```

Ответ **green** будет признаком корректного функционирования Elasticsearch

3.8.4 Методика контроля параметров Kibana

Проверка проводится в командной строке на хостах с ролью **log**

```
docker exec -t fluentd-server curl -svo /dev/null --user
"$KIBANA_USER:$KIBANA_PASS" "http://$DOCKER_IP/app/kibana/" | grep '<
HTTP' | awk '{ print $4 }'
```

Ответ **OK** будет признаком корректного функционирования Kibana

4 ПОЧТА (PSN)

В описании методов контроля ниже используются переменные вида **$fe_host_mail**, которые определяют параметры конкретного окружения. Значения переменных можно получить из файла inventory Список переменных:

- **$fe_mail_host-** определяет IP адреса хостов, на котором развернуты frontend сервисы Почты PSN (роль FE)
- **$be_mail_host-** определяет IP адреса хостов, на котором развернуты backend сервисы Почты PSN (роль BE)

4.1 Соответствие между описаниями Почта (PSN)

Таблица, приведенная ниже, устанавливает соответствие между описанием сервисов Почта (PSN) в текущем разделе документа и ролями хостов, описанными в разделе 4.2.1 документа **Руководство по установке. МойОфис Почта**.

Таблица 11. Таблица соответствия. Указатель

Сервис	Сервисно-ресурсная модель	Роль(группа хостов МойОфис Почта)
MariaDB Service	Подсистема база данных	db(db)
IPVS Service	Подсистема балансировки	keepalived(lb)
Rspamd Service	Подсистема контент-контроля	be(be)
ClamAV Service	Подсистема контент-контроля	be(be)
Dovecot Service	Подсистема доставки сообщений	be(be)
Postfix Service	Подсистема транспорта сообщений	be(be)
WFE Service	Подсистема интерфейса пользователя	webclient(fe)
Nginx Service	Подсистема интерфейса пользователя	
LDAP Service	Подсистема авторизации	ldap(userdb)
LDAP Service	Подсистема глобальной адресной книги	ldap(cab)
PSN API Service	Подсистема управления	psnapi(fe)
Autoconfig Service	Подсистема управления	ac(fe)
Poseidon Backend Manager Service	Подсистема управления	be(be)
Poseidon Frontend Manager Service	Подсистема управления	fe(fe)

4.2 Подсистема база данных

Таблица 12. Параметры Подсистемы база данных

Контролируемый параметр	Корректное значение	Критичность
"MariaDB Service"	"Up"	Critical

4.2.1 Методика контроля MariaDB service

Для проверки статуса MariaDB service на хостах BE с ролью выполнить команду:

```
docker ps --format "table {{.Names}}\t{{.Status}}" | grep mariadb
```

4.3 Подсистема балансировки

Таблица 13. Параметры Подсистемы балансировки

Контролируемый параметр	Корректное значение	Критичность
"IPVS Service"	"Непустой ответ"	Critical

4.3.1 Методика контроля IPVS Service

Проверка статуса работы IPVS Service осуществляется в командной строке на хостах с ролью **LB**:

```
ipvsadm -Ln
```

Пример ответа:

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP 10.160.115.170:25 wlc persistent 5
-> 10.160.115.154:25 Route 50 0 0
-> 10.160.115.155:25 Route 50 0 0
TCP 10.160.115.170:80 wlc persistent 5
-> 10.160.115.154:80 Route 50 0 0
-> 10.160.115.155:80 Route 50 0 0
```

```
TCP 10.160.115.170:443 wlc persistent 5
-> 10.160.115.154:443 Route 50 24 15
-> 10.160.115.155:443 Route 50 7 14
TCP 10.160.115.170:587 wlc persistent 5
-> 10.160.115.154:587 Route 50 0 0
-> 10.160.115.155:587 Route 50 0 0
TCP 10.160.115.170:8080 wlc persistent 5
-> 10.160.115.154:8080 Route 50 64 15
-> 10.160.115.155:8080 Route 50 72 14
```

4.4 Подсистема контент-контроля

Таблица 14. Параметры Подсистемы контент-контроля

Контролируемый параметр	Корректное значение	Критичность
"Rspamd Service"	"RUNNING"	Critical
"ClamAV Service"	"RUNNING"	Critical

4.4.1 Методика контроля Rspamd Service

Для проверки статуса Rspamd Service на хостах с ролью BE выполнить команду:

```
docker exec mail-2020-02 supervisorctl status rspamd
```

4.4.2 Методика контроля ClamAV Service

Для проверки статуса ClamaAV Service на хостах с ролью BE выполнить команду:

```
docker exec mail-2020-02 supervisorctl status clamav
```

4.5 Подсистема доставки сообщений

Таблица 15. Параметры Подсистемы доставки сообщений

Контролируемый параметр	Корректное значение	Критичность
"Dovecot Service"	"RUNNING"	Critical

4.5.1 Методика контроля Dovecot Service

Проверка статуса Dovecot Service на хостах с ролью **BE** выполнить команду:

```
docker exec -i mail-2020-02 supervisorctl status dovecot
```

Дополнительно состояние сервиса (ответ сервиса на запрос) может быть проверено запросом в командной строке на узле с развернутым сервисом Dovecot:

```
docker exec -i mail-2020-02 echo QUIT | nc localhost 110
```

Пример ответа:

```
+OK Dovecot ready.
+OK Logging out
```

4.6 Подсистема транспорта сообщений

Таблица 16. Параметры Подсистемы транспорта сообщений

Контролируемый параметр	Корректное значение	Критичность
"Postfix Service"	"RUNNING"	Critical

4.6.1 Методика контроля Postfix Service

Проверка статуса Postfix Service на хостах с ролью BE выполнить команду:

```
docker exec -i mail-2020-02 supervisorctl status postfix:master
```

Дополнительно состояние сервиса (ответ сервиса на запрос) может быть проверено запросом в командной строке на узле с развернутым сервисом Postfix:

```
docker exec -i mail-2020-02 echo QUIT | nc localhost 25
```

Пример ответа:

```
220 host.domain ESMTP host.domain
221 2.0.0 Bye
```

4.7 Подсистема интерфейса пользователя

Таблица 17. Параметры Подсистемы интерфейса пользователя

Контролируемый параметр	Корректное значение	Критичность
"WFE Service"	Любой ответ, кроме сообщения об ошибке	Warning
"Nginx Service"	"RUNNING"	Critical

4.7.1 Методика контроля WFE Service

Для проверки статуса WFE Service выполнить команду:

```
curl -s https://$fe_mail_host/version | jq '.psnapi[0].version'
```

Проверка должна проводится с хоста, который находится в одной локальной сети с хостом **$fe_mail_host**.

Пример ответа:

```
"0.1.3"
```

Ответ не должен содержать сообщения об ошибке.

4.7.2 Методика контроля Nginx Service

Для проверки статуса Nginx Service на хостах с ролью FE выполнить команду:

```
docker exec psn-2020-02 supervisorctl status nginx
```

4.8 Подсистема авторизации

Таблица 18. Параметры Подсистемы авторизации

Контролируемый параметр	Корректное значение	Критичность
"LDAP Service"	"UP"	Critical

4.8.1 Методика контроля LDAP Service

Проверка статуса сервиса LDAP Service на хостах с ролью **BE** выполнить команду:

```
docker ps --format "table {{.Names}}\t{{.Status}}" | grep ldap
```

Дополнительная проверка (ответ сервиса на запрос) может быть осуществлена следующим запросом в командной строке на узле с развернутым сервисом LDAP:

```
docker exec ldap-2020-02 ldapsearch -x -LLL -b 'cn=Monitor' –s base
'(objectClass=*)' '@monitorServer'
```

Пример ответа:

```
dn: cn=monitor
```

4.9 Подсистема глобальной адресной книги

Таблица 19. Параметры Подсистемы глобальной адресной книги

Контролируемый параметр	Корректное значение	Критичность
"LDAP Service"	"UP"	Minor

4.9.1 Методика контроля LDAP Service

Проверка статуса работы подсистемы LDAP Service осуществляется аналогично разделу Подсистема авторизации, но на хосте, на котором развернут сервис глобальной адресной книги.

4.10 Подсистема управления

Таблица 20. Параметры Подсистемы управления

Контролируемый параметр	Корректное значение	Критичность
"PSN API Service"	"200 OK"	Critical
"Autoconfig Service"	Любой ответ, не содержащий сообщения об ошибке	Warning
"Poseidon Backend Manager Service"	RUNNING	Major
"Poseidon Frontend Manager Service"	RUNNING	Major

4.10.1 Методика контроля PSN API Service

Проверка статуса работы подсистемы осуществляется http-запросом в браузере или в командной строке. Проверка должна проводится с хоста, который находится в одной локальной сети с хостом **$fe_mail_host**:

```
curl -k 'https://$fe_mail_host:666/users?q=test@myoffice.ru'
```

Пример ответа в случае штатной работы сервиса PSN API Service

```
{ "results": ["test@myoffice.team"] }
```

4.10.2 Методика контроля Autoconfg Service

Проверка статуса работы подсистемы осуществляется http(s)-запросом в браузере или в командной строке:

```
curl –s
'https://$fe_mail_host:444/?login=test@myoffice.ru&password=test_pass'
```

где

test@myoffice.ru - логин тестового пользователя

test_pass - пароль тестового пользователя

Признаком корректной работы сервиса Autoconfg Service служит получение в ответ непустого ответа, содержащего набор параметров конфигурации в формате json.

4.10.3 Методика контроля Poseidon Backend Manager Service

Проверка статуса работы подсистемы осуществляется в командной строке на узлах с ролью PSN Backend(be):

```
docker exec mail-2020-02 supervisorctl status backend-manager
```

Дополнительная проверка:

```
docker exec -i mail-2020-02 echo QUIT | nc localhost 48666
```

Любой вывод команды, кроме 501 Internal Server Error

Пример возможного ответа

```
HTTP/1.1 400 Bad Request
Server: nginx/1.16.1
Date: Tue, 24 Mar 2020 15:54:44 GMT
Content-Type: text/html
Content-Length: 157
Connection: close
<html>
<head><title>400 Bad Request</title></head>
```

```
<body>
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx/1.16.1</center>
</body>
</html>
```

4.10.4 Методика контроля Poseidon Frontend Manager Service

Проверка статуса работы подсистемы осуществляется в командной строке на узлах с ролью PSN Frontend(fe):

```
docker exec -i psn-2020-02 supervisorctl status frontend-manager
```

5 МЕССЕНДЖЕР (LOGOS)

Штатным вариантом развертывания продукта Мессенджер(Logos) является развертывание с помощью Docker. Такой вариант развертывания подразумевает запуск каждого сервиса из состава Мессенджер(Logos) в отдельном контейнере. Поэтому для некоторых сервисов статус отдельного сервиса проверяется как статус контейнера, в котором исполняется соответствующий сервис. Однако, проверки только статуса контейнера может быть недостаточно. Описание двух дополнительных проверок на примере сервиса **nginx** приведено ниже.

Во-первых, следует убедиться, что контейнер не находится в статусе перезагрузки (команда вернет **false**):

```
docker inspect -f '{{.State.Restarting}}' $(docker ps | grep "
logos_web_nginx" | awk '{ print $1 }')
```

Во-вторых, контейнер не был перезапущен с момента предыдущей проверки (команда вернет время запуска контейнера):

```
docker inspect -f '{{.State.StartedAt}}' $(docker ps | grep "
logos_web_nginx" | awk '{ print $1 }')
```

В штатном режиме время, прошедшее с момента запуска контейнера, будет больше времени, прошедшего с момента предыдущей проверки.

Для сервисов, входящих в состав Подсистемы API, Подсистемы авторизации, Подсистемы выполнения фоновых задач и Подсистемы связи с клиентскими приложениями, проверки организуются иначе. Каждый из сервисов в ответ на запрос на определенный порт сообщает свой статус. Например, статус сервиса **vega** Подсистемы авторизации проверяется так:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+vega:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//' -e
's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_http --json -H http://vega:9020 | grep
alive
```

Пример ответа:

```
"alive": true
```

Список сервисов, отвечающих на запросы по протоколу **http**, можно получить командой на любом хосте с Logos:

```
cat /opt/co-logos/swarm/stacks/logos-backend.yml | grep addr -B 1
```

Список сервисов, отвечающих на запросы по протоколу **grpc**, можно получить командой на любом хосте с Logos:

```
cat /opt/co-logos/swarm/stacks/logos-backend.yml | grep listen -B 1
```

5.1 Подсистема База данных

Таблица 21. Параметры Подсистемы База данных

Контролируемый параметр	Корректное значение	Критичность
"PostgreSQL Service"	"running"	Critical

5.1.1 Методика контроля PostgreSQL Service

Проверка статуса работы PostgreSQL Service осуществляется в командной строке:

```
docker inspect -f '{{.State.Status}}' $(docker ps | grep "postgres" | awk '{ print $1 }')
```

Пример ответа:

```
running
```

5.2 Подсистема авторизации

Таблица 22. Параметры Подсистемы авторизации

Контролируемый параметр	Корректное значение	Критичность
"vega service"	""alive": true"	Critical

5.2.1 Методика контроля Vega Service

Проверка статуса работы **Vega Service** осуществляется в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+vega:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//' -e
's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_http --json -H http://vega:9020 | grep
alive
```

5.3 Подсистема выполнения фоновых задач

Таблица 23. Параметры Подсистемы выполнения фоновых задач

Контролируемый параметр	Корректное значение	Критичность
"aquarii service"	""alive": true"	Critical

5.3.1 Методика контроля aquarii service

Проверка статуса работы **aquarii service** осуществляется в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+aquarii:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//'
-e 's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_http --json -H http://aquarii:9040 |
grep alive
```

5.4 Подсистема связи с клиентскими приложениями

Таблица 24. Параметры Подсистемы связи с клиентскими приложениями

Контролируемый параметр	Корректное значение	Критичность
"maia service"	""alive": true""	Critical
"altair service"	""alive": true"	Critical
"rigel service"	""alive": true"	Critical

5.4.1 Методика контроля maia service

Проверка статуса работы maia service осуществляется в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+maia:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//' -e
's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_grpc --json -ca=./cert/ca.pem -client-
cert=./cert/cert.pem -client-key=./cert/key.pem -tls-host-
override=logos.dev -H maia:50053 | grep alive
```

5.4.2 Методика контроля altair service

Проверка статуса работы altair service осуществляется в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+altair:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//' -
e 's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_http --json -H http://altair:9030 | grep
alive
```

5.4.3 Методика контроля rigel service

Проверка статуса работы rigel service осуществляется в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+rigel:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//' -e
's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_grpc --json -ca=./cert/ca.pem -client-
cert=./cert/cert.pem -client-key=./cert/key.pem -tls-host-
override=logos.dev -H rigel:50055 | grep alive
```

5.5 Подсистема API

Таблица 25. Параметры Подсистемы API

Контролируемый параметр	Корректное значение	Критичность
"regulus service"	""alive": true""	Critical
"mizar service"	""alive": true""	Critical
"taygeta service"	""alive": true""	Critical
"sirius service"	""alive": true""	Critical
"alcor service"	"running"	Critical

5.5.1 Методика контроля regulus service

Проверка статуса работы regulus service осуществляется командой в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+regulus:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//'
-e 's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_http --json -H http://regulus:9025 |
grep alive
```

5.5.2 Методика контроля mizar service

Проверка статуса работы mizar service осуществляется командой в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+mizar:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//' -e
's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_http --json -H http://mizar:9015 | grep
alive
```

5.5.3 Методика контроля taygeta service

Проверка статуса работы taygeta service осуществляется командой в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+taygeta:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//'
-e 's/"$//'`
```

```
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_grpc --json -ca=./cert/ca.pem -client-
cert=./cert/cert.pem -client-key=./cert/key.pem -tls-host-
override=logos.dev -H taygeta:50054 | grep alive
```

5.5.4 Методика контроля sirius service

Проверка статуса работы sirius service осуществляется командой в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+sirius:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//' -
e 's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_grpc --json -ca=./cert/ca.pem -client-
cert=./cert/cert.pem -client-key=./cert/key.pem -tls-host-
override=logos.dev -H sirius:50051 | grep alive
```

5.5.5 Методика контроля alcor service

Проверка статуса работы alcor service осуществляется командой в командной строке:

```
export LOGOS_IMAGE=`cat /opt/co-logos/swarm/stacks/logos-backend.yml |
grep -E -A 1 "^\s+alcor:" | tail -n 1 | sed -e 's/\s\+image:\s\+"\?//' -e
's/"$//'`
docker run -it --rm --network=co-net -v /root/.docker:/opt/logos/cert
${LOGOS_IMAGE} ./bin/check_logos_grpc --json -ca=./cert/ca.pem -client-
cert=./cert/cert.pem -client-key=./cert/key.pem -tls-host-
override=logos.dev -H alcor:50052 | grep alive
```

5.6 Подсистема обслуживания очередей

ВНИМАНИЕ	Конфигурация Подсистемы в автономной и кластерной конфигурации отличается. В случае кластерной конфигурации используются сервисы из состава компонента Редакторы (CO).

Таблица 26. Подсистема обслуживания очередей

Контролируемый параметр	Корректное значение	Критичность
"Redis service"	"running"	Critical
"RabbitMQ service"	"running"	Critical

5.6.1 Методика контроля Redis service в случае развертывания в автономной конфигурации

Проверка статуса работы Redis service осуществляется в командной строке:

```
docker inspect -f '{{.Status.State}}' $(docker stack ps -q redis)
```

5.6.2 Методика контроля Redis service в случае развертывания в кластерной конфигурации

Проверка статуса работы Redis service осуществляется в командной строке:

```
docker inspect -f '{{.Status.State}}' $(docker stack ps -q redis-sentinel)
```

5.6.3 Методика контроля RabbitMQ service в случае развертывания в автономной конфигурации

Проверка статуса работы RabbitMQ service осуществляется в командной строке:

```
docker inspect -f '{{.Status.State}}' $(docker stack ps -q mq)
```

5.6.4 Методика контроля RabbitMQ service в случае развертывания в кластерной конфигурации

Методика проверки описана в разделе **Методика контроля RabbitMQ Cluster**

5.7 Подсистема балансировки

Таблица 27. Параметры Подсистемы балансировки

Контролируемый параметр	Корректное значение	Критичность
"Nginx Service"	"running"	Critical

5.7.1 Методика контроля Nginx Service

Проверка статуса работы Nginx Service осуществляется командой в командной строке:

```
docker inspect -f '{{.State.Status}}' $(docker ps | grep
"logos_web_nginx" | awk '{ print $1 }')
```

5.8 Подсистема Web интерфейса

Таблица 28. Параметры Подсистемы Web интерфейса

Контролируемый параметр	Корректное значение	Критичность
"mira Application"	"running"	Critical
"GOST TLS termination service"	"running"	Critical

5.8.1 Методика контроля mira Application

Проверка статуса работы mira Application осуществляется в командной строке:

```
docker inspect -f '{{.State.Status}}' $(docker ps | grep "mira" | awk '{
print $1 }')
```

Ожидаемое значение:

```
running
```

5.8.2 Методика контроля GOST TLS termination service

ВНИМАНИЕ	Данная проверка распространяется только на версии Частного Облака, содержащие криптографические преобразования согласно ГОСТ РФ

Проверка статуса работы **GOST TLS termination service** осуществляется в командной строке:

```
docker inspect -f '{{.State.Status}}' $(docker ps | grep "nginx-gost" |
awk '{ print $1 }')
```

Ожидаемое значение:

```
running
```

6 ХРАНИЛИЩЕ PGS

6.1 Описание сервисов

Посмотреть все сервисы и их статус можно командой:

```
docker service ls --format 'table {{.Name}}\t{{.Replicas }}'
```

Для релиза 2020.02.R2 по умолчанию поддерживается инсталляция с количеством реплик = 1, соответственно вывод должен быть следующим:

```
NAME                                REPLICAS
pgs-arangodb_arangodb               1/1
pgs-elasticsearch_elasticsearch     1/1
pgs-etcd_etcd                       1/1
pgs-keycloak_keycloak               1/1
pgs-keycloak_postgres               1/1
pgs-rabbitmq_rabbitmq               1/1
pgs-redis_redis                     1/1
pgs_aristoteles                     1/1
pgs_euclid                          1/1
pgs_polemon                         1/1
pgs_sisyphussearch                  1/1
pgs_sisyphusworker                  1/1
```

6.2 Подсистема Ядро API

Таблица 29. Параметры Подсистемы Ядро API

Контролируемый параметр	Корректное значение	Критичность
"Aristoteles"	"Running"	Critical
"Redis Service"	"Running"	Critical

6.2.1 Методика контроля Aristoteles

Проверка статуса работы Aristoteles Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}'
pgs_aristoteles
```

Пример ответа:

```
NAME DESIRED STATE
pgs_aristoteles.1 Running
```

6.2.2 Методика контроля Redis Service

Проверка статуса работы Redis Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' pgs-
redis_redis
```

Пример ответа:

```
NAME                DESIRED STATE
pgs-redis_redis.1   Running
```

6.3 Подсистема управления пользователями

Таблица 30. Параметры Подсистемы управления

Контролируемый параметр	Корректное значение	Критичность
"Keycloak"	"Running"	Critical
"PostgreSQL"	"Running"	Critical

6.3.1 Методика контроля Keycloak Service

Проверка статуса работы Keycloak Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' pgs-
keycloak_keycloak
```

Пример ответа:

```
NAME                      DESIRED STATE
pgs-keycloak_keycloak.1   Running
```

6.3.2 Методика контроля Postgres Service

Проверка статуса работы Keycloak Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' pgs-pgs-keycloak_postgres
```

Пример ответа:

```
NAME                      DESIRED STATE
pgs-keycloak_postgres.1   Running
```

6.4 Подсистема поиска

Таблица 31. Параметры Подсистемы "Поиск"

Контролируемый параметр	Корректное значение	Критичность
"Elasticsearch"	"Running"	Critical
"RabbitMQ"	"Running"	Critical
"SisyphusSearch"	"Running"	Critical
"SisyphusWorker"	"Running"	Major

6.4.1 Методика контроля Elasticsearch Service

Проверка статуса работы Elasticsearch Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' pgs-elasticsearch_elasticsearch
```

Пример ответа:

```
NAME                                DESIRED STATE
pgs-elasticsearch_elasticsearch.1   Running
```

6.4.2 Методика контроля RabbitMQ Service

Проверка статуса работы RabbitMQ Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' pgs-rabbitmq_rabbitmq
```

Пример ответа:

```
NAME                      DESIRED STATE
pgs-rabbitmq_rabbitmq.1   Running
```

6.4.3 Методика контроля SisyphusSearch Service

Проверка статуса работы SisyphusSearch Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}'
pgs_sisyphussearch
```

Пример ответа:

```
NAME                   DESIRED STATE
pgs_sisyphussearch.1   Running
```

6.4.4 Методика контроля SisyphusWorker Service

Проверка статуса работы SisyphusWorker Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}'
pgs_sisyphusworker
```

Пример ответа:

```
NAME                    DESIRED STATE
pgs_sisyphusworker.1    Running
```

6.5 Подсистема хранения метаданных файлов и прав доступа

Таблица 32. Параметры Подсистемы хранения метаданных файлов и прав доступа"

Контролируемый параметр	Корректное значение	Критичность
"ArangoDB"	"Running"	Critical

6.5.1 Методика контроля ArangoDB Service

Проверка статуса работы ArangoDB Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' pgs-
arangodb_arangodb
```

Пример ответа:

```
NAME                        DESIRED STATE
pgs-arangodb_arangodb.1     Running
```

6.6 Подсистема администрирования

Таблица 33. Параметры Подсистемы администрирования

Контролируемый параметр	Корректное значение	Критичность
"Euclid"	"Running"	Critical
"Polemon"	"Running"	Major

6.6.1 Методика контроля Euclid Service

Проверка статуса работы Euclid Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}'
pgs_euclid
```

Пример ответа:

```
NAME           DESIRED STATE
pgs_euclid.1   Running
```

6.6.2 Методика контроля Polemon Service

Проверка статуса работы Polemon Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}'
pgs_polemon
```

Пример ответа:

```
NAME            DESIRED STATE
pgs_polemon.1   Running
```

6.7 Подсистема конфигурирования

Таблица 34. Параметры Подсистемы конфигурирования

Контролируемый параметр	Корректное значение	Критичность
"etcd"	"Running"	Critical

6.7.1 Методика контроля ETCD Service

Проверка статуса работы ETCD Service сервиса в командной строке:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' pgs-etcd_etcd
```

Пример ответа:

```
NAME              DESIRED STATE
pgs-etcd_etcd.1   Running
```