МОДЕЛИРОВАНИЕ СИСТЕМ

УДК 004

РАЗРАБОТКА МОДУЛЯ АВТОМАТИЗАЦИИ ОБРАБОТКИ ИНЦИДЕНТОВ ДЛЯ СИСТЕМ SERVICE DESK

© 2022 Н. В. Баранов, А. Н. Зеленина

Воронежский институт высоких технологий (Воронеж, Россия)

Представлена разработка модуля автоматизации обработки инцидентов для систем Service Desk – несколько бекэнд-служб (ботов), работающих на Linux-сервере. Планируемая область применения – провайдер или аутсорс-компания, занимающаяся обработкой инцидентов в системах Service Desk. К достоинствам планируемого решения автоматизации, по сравнению с аналогами, относится: сервисы (боты) автоматически проверяют актуальность задач, обрабатывают типичные рутинные задачи, составляют и вносят отчёт в инцидент, а также назначают ответственных. Это позволит избавить инженеров мониторинга и отдела устранения неисправностей от многократного повторения рутинных задач при обработке типичных инцидентов.

Ключевые слова: обработкой инцидентов в системах Service Desk, сервис (бот).

Введение

По мере развития сетей мобильной связи и интернет, постоянно растёт количество сетевых устройств. На территории России эксплуатируется более 2 млн. радио-электронных систем различных служб радиосвязи [1]. К тому же наблюдается тенденция, что при увеличении скорости снижается максимальное расстояние передачи, что в свою очередь увеличивает количество устройств передатчиков (рис. 1).

Силами отделов строительства в зоне ответственности АО «НСН» каждый год строится около двух тысяч новых позиций, а в прошлом году только в г. Москва было построено больше тысячи. Количество обслуживаемых устройств растёт, а следовательно, растёт количество регистрируемых аварий на этих устройствах, а для каждой аварии заводится тикет в системе Service Desk. Следовательно работы у отделов мониторинга и устранения неисправностей становится, буквально с каждым днём, всё больше. Поэтому было принято решение о разработке модуля автоматизации для обработки и диспетчеризации однотипных, рутинных задач.

Разработка модуля автоматизации. Основным языком разработки выбран Python – как язык с простым синтаксисом, и большим количеством сторонних библиотек, облегчающих разработку.

Для хранения данных предполагалось использовать базу данных на PostgreSQL, основными плюсами которой является простота работы как с обычной SQL базой и возможность хранить базу на сервере в сети. Но, в процессе разработки, от базы было решено отказаться, так как сервисы все равно взаимодействуют с базой данный системы IMS (Incident Management System), а хранить одну и ту же информацию – излишне, при условии, что данные в IMS могут изменяться, сохранённая копия может стать неактуальной.

Для управления версиями используется система контроля версий Git – абсолютный лидер по популярности среди современных систем управления версиями, производительный, быстрый, гибкий.

Баранов Никита Викторович – Воронежский институт высоких технологий, студент.

Зеленина Анна Николаевна – Воронежский институт высоких технологий, канд. техн. наук, доцент, e-mail: snakeans@gmail.com.

Работают боты как systemd службы на виртуальном сервере под управлением системы Ubuntu Server. Достоинства Ubuntu – это безопасность, простота обслуживания, а также нативная поддержка Python.

Выбранные средства разработки представлены на рисунке 2.

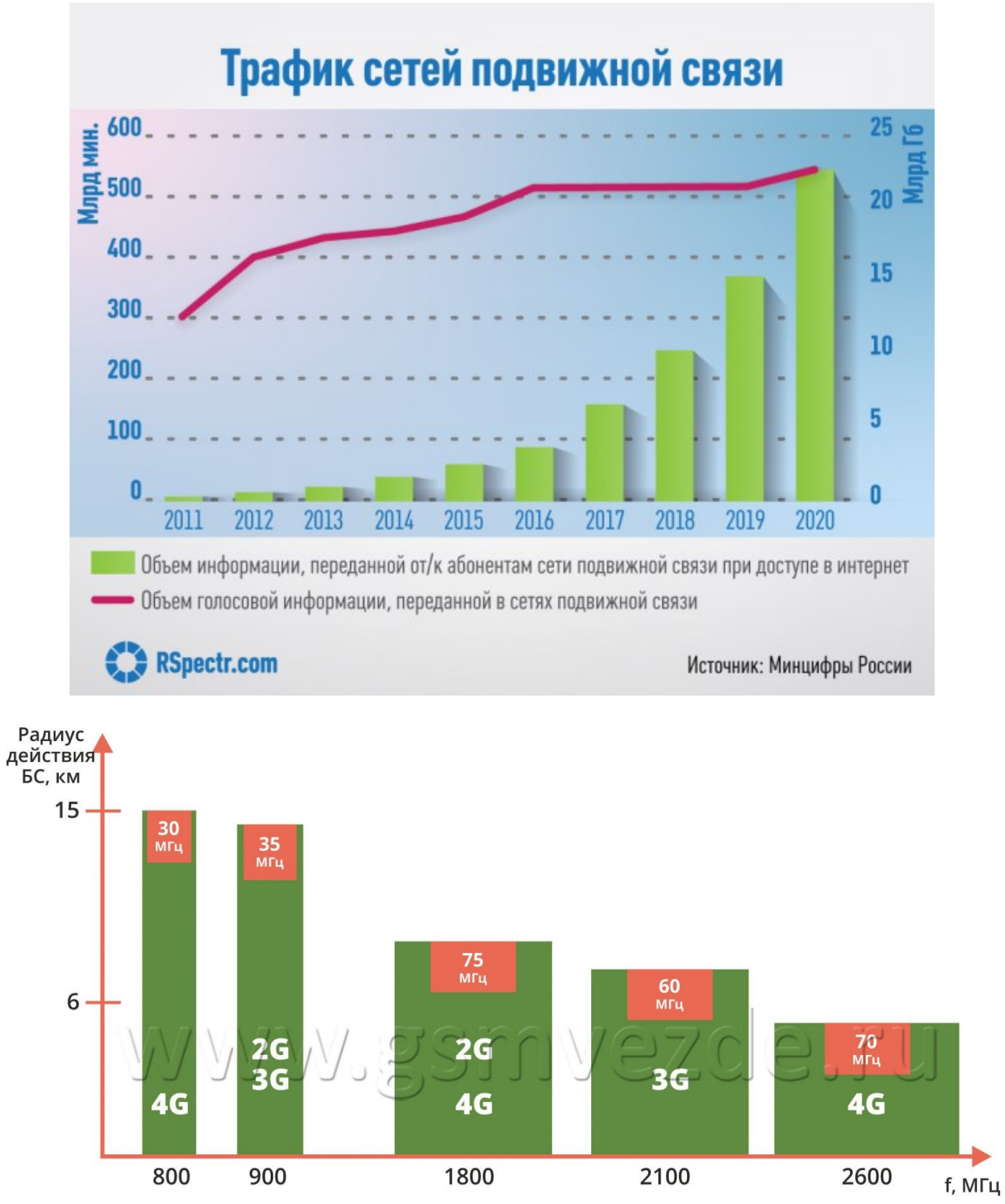

Рисунок 1. Тенденция развития сетей мобильной связи и интернет: динамика роста трафика мобильной сети и радиус покрытия базовых станций разных поколений и частот [2, 3]

Рисунок 2. Выбранные средства разработки модуля автоматизации

Совместно с инженерами отделов мониторинга и устранения неисправностей были разработаны алгоритмы обработки и диспетчеризации типовых инцидентов.

Бот 1. Каждые 2 минуты он запрашивает данные из системы IMS, отбирает из них инциденты 3 и 4 приоритетов, назначенные на группы AM (Alarm Monitoring), FM (Fault Management), FM-TN (Fault Management Transport Network), территориально привязанные к макрорегионам Центр и Юг, содержащие ряд стандартных заголовков и комментариев. И «берёт» их в работу, назначает на специально зарегистрированного в системе IMS исполнителя.

Затем вступает второй бот и распределяет инциденты на списки по макрорегионам и регионам, а затем для каждого из 32 списков регионов инциденты делятся на 2 группы: первая для инцидентов, которые не имеют активной аварии по отсутствию канала управления (на рисунке 3 отмечена цифрой 1.0) и далее обрабатывается на предмет аварий КСВ и/или перегрева, вторая для тех, что имеют аварию по каналу управления.

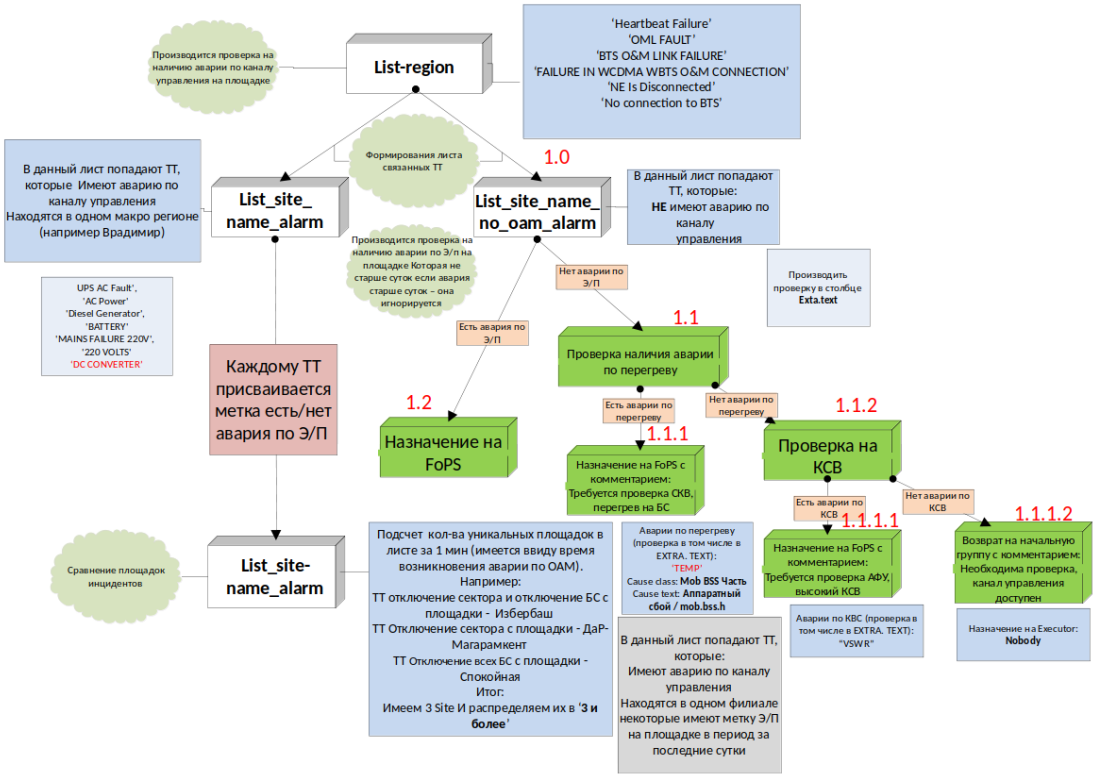

Рисунок 3. Часть алгоритма второго бота, отвечающего за обработку инцидентов

Инциденты с авариями по каналу управления обрабатываются практически одинаково в зависимости от количество базовых станций в основном назначаются на группы регионов. На рисунке 4 видим схему для макрорегиона Центр.

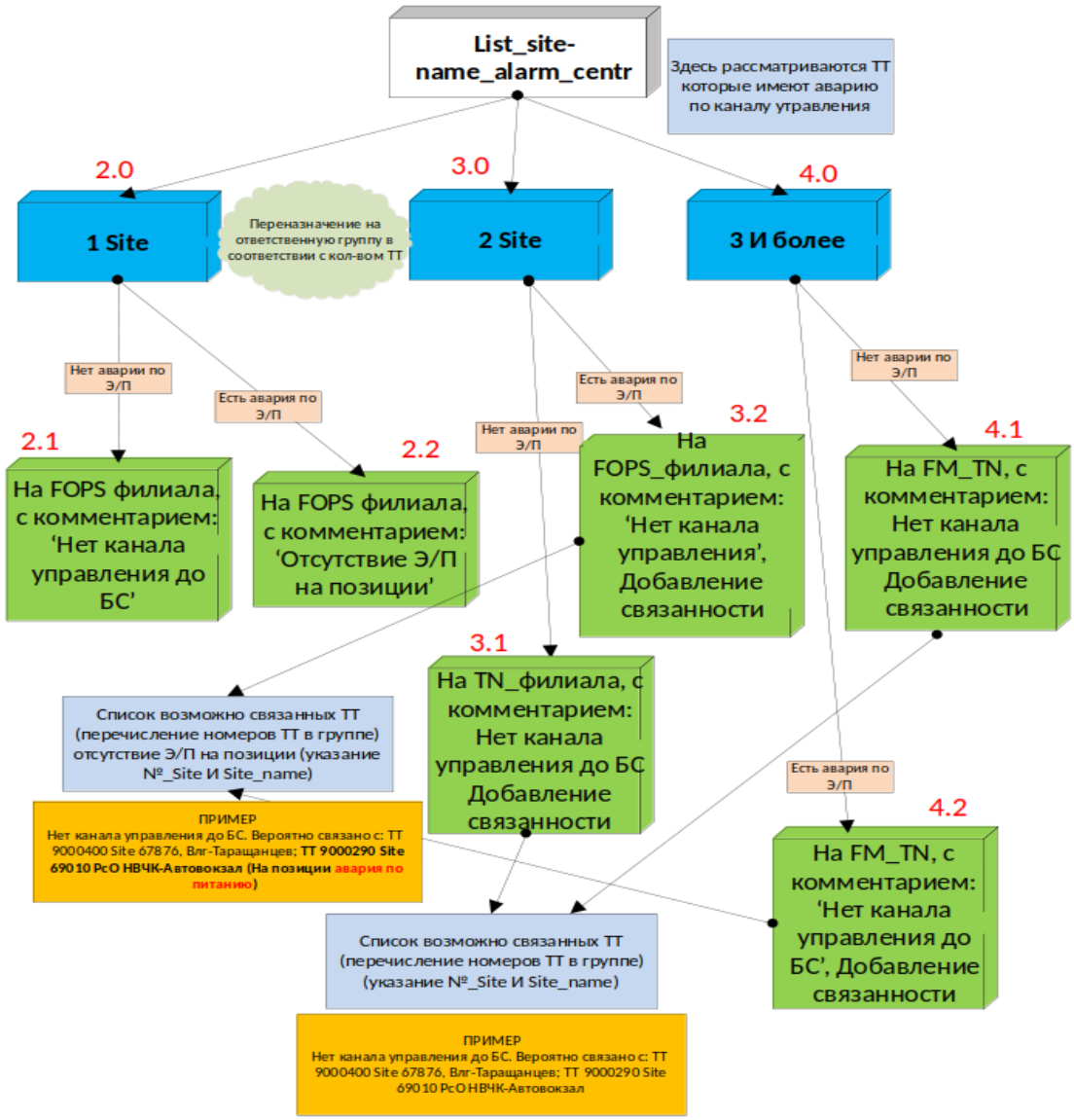

Рисунок 4. Часть алгоритма второго бота, отвечающего за обработку инцидентов макрорегиона Центр

На рисунке 5 помимо местной эксплуатации появляется отдел FM TN (Fault Management Transport Network) GDC, обрабатывающий инциденты на транспортной сети.

После разработки алгоритмов составлена приблизительная модель программных модулей и нужных сторонних библиотек. Созданы виртуальное окружение с помощью утилиты poetry и git репозиторий с помощью консольной утилиты git.

Для выполнения задачи было решено создать двух ботов на языке python, для каждого из них был создан свой модуль, main_dispatcher_fm_bot_1_1.py и main_dispatcher_fm_bot_1_2.py. В ходе разработки для каждого из них был собран список одинаковых параметров, было решено вынести их в конфигурационный файл, который в итоге разделился на два отдельных файла для каждого из ботов config_dispatcher_fm_bot_1_1.py и

config_dispatcher_fm_bot_1_2.py, так как значений для некоторых параметров отличались, а так же туда были добавлены константы уникальные для каждого из ботов.

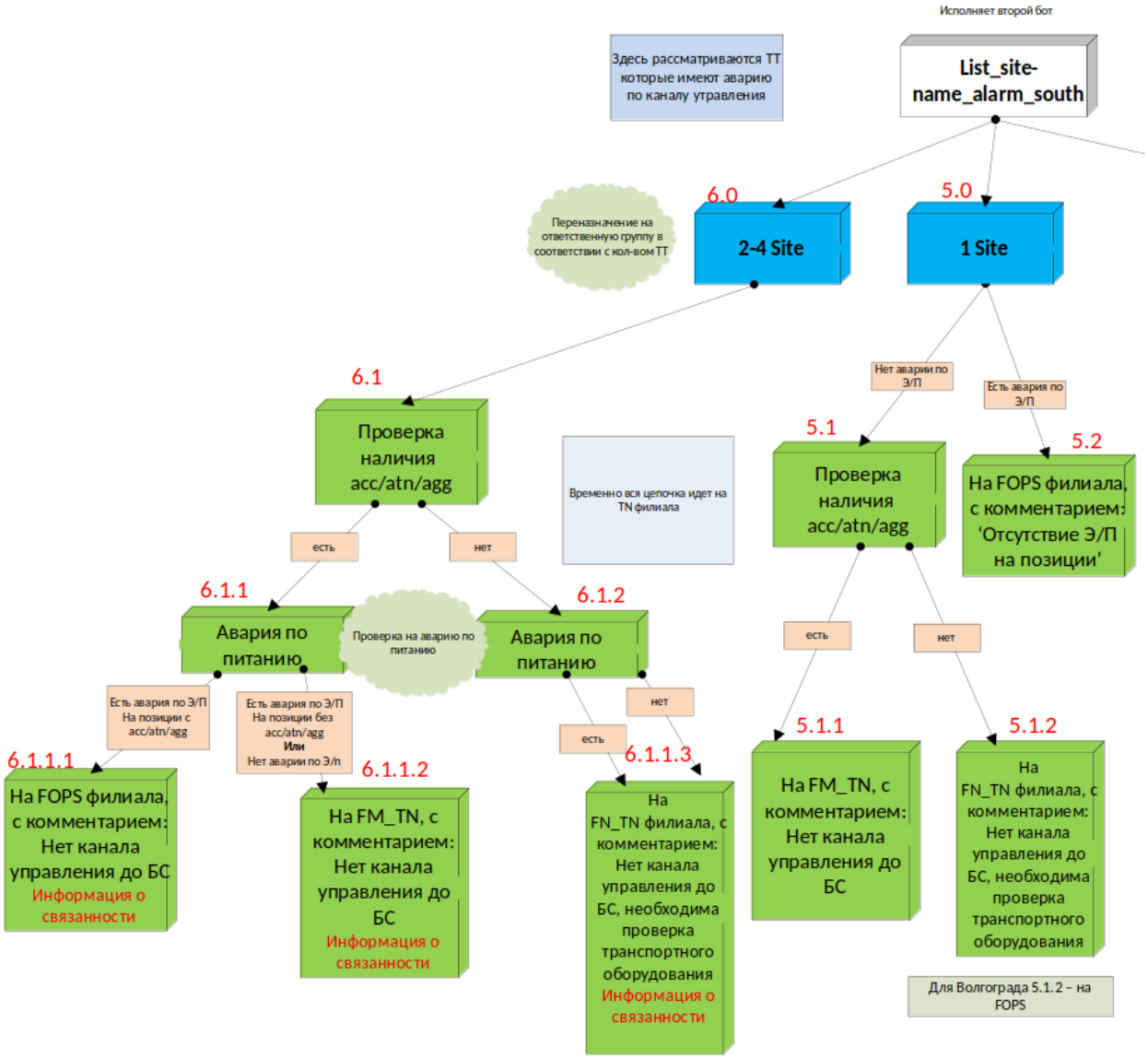

Рисунок 5. Часть алгоритма второго бота, отвечающего за обработку инцидентов микрорегиона ЮГ

После разработки алгоритмов составлена приблизительная модель программных модулей и нужных сторонних библиотек. Созданы виртуальное окружение с помощью утилиты poetry и git репозиторий с помощью консольной утилиты git.

Для выполнения задачи было решено создать двух ботов на языке python, для каждого из них был создан свой модуль, main_dispatcher_fm_bot_1_1.py и main_dispatcher_fm_bot_1_2.py. В ходе разработки для каждого из них был собран список одинаковых параметров, было решено вынести их в конфигурационный файл, который в итоге разделился на два отдельных файла для каждого из ботов config_dispatcher_fm_bot_1_1.py и config_dispatcher_fm_bot_1_2.py, так как значений для некоторых параметров отличались, а так же туда были добавлены константы уникальные для каждого из ботов.

Для первого:

- Ключевые слова в поле заголовок для поиска инцидентов;
- Ключевые слова в поле Комментарий для поиска инцидентов;
- Id центральных регионов;
- Id южных регионов.

Для второго:

- Ключевые слова для поиска аварий на площадке
- Id групп FoPS разных регионов;
- Id групп FN_TN и TN разных регионов.

Так же в отдельный файл authentication.py была вынесена информация для подключения к разным ресурсам:

- Базе данных Postgres, для хранения токенов ботов;
- REST_API NSN Portal для взаимодействия с Service Desk системой на портале;
- Telegram bot API для оповещения команды разработки о критических сбоях в работе ботов.

При составлении запросов к API NSN Portal, чтобы соответствовать принципу программирования не повторяй себя были созданы шаблоны для запросов и вынесены в отдельный файл http_requests.py, чтобы потом их можно было легко импортировать и использовать в разных ботах, просто заполняя недостающими данными. Были созданы следующие шаблонные методы

- rest_api – для работы с REST API NSN портала.
- check_rest – для проверки доступности REST API NSN портала.
- telegram – для работы с API Telegram.

Так же в отдельный файл nsn_incident.py была вынесена вся, так называемая, бизнес логика взаимодействия с NSN Portal. Создан класс FormationInfo для хранения данные, полученные из запроса от NSN портала, а также следующие методы:

- formation_geography_incidents() – для разделения инцидентов по макрорегионам;
- edit_group_id_in_incidents_for_reassignment_by_region() – для изменения id группы в инциденте при подготовке его к переназначению, соответствующий региону;
- insert_group_id_in_incidents – для вставки id группы в инциденте при подготовке его к переназначению;
- insert_cause_in_incident() – для добавления причины переназначения;
- formation_info_by_type_reassign_id_in_list_incidents_for_reassignment() – формирование информации о по типу переназначения в списке инцидентов для переназначенияю

Класс FiltrationInfo наследуемый от FormationInfo и содержащий следующие методы:

- filter_by_title() – фильтрация по заголовку;
- filter_by_updated_at() – фильтрация по дате обновления инцидентах;
- filter_by_comment() – фильтрация по комментарию;
- filter_by_date_of_creation() – фильтрация по дате создания;
- filter_by_main_site_id() – фильтрация по id главное позиции в инциденте;
- filter_by_network_element_id() – фильтрация по сетевому элементу;
- filter_by_relationships() – фильтрация инцидентов по отношения, зависимый/главный;
- filter_by_executor() – фильтр по исполнителю;
- list_incidents_with_title_4G() – фильтрация инцидентов с признаками 4G в заголовке.

Класс AnalysisInfoFmBot1 наследуемый от FiltrationInfo, который содержит следующие методы:

- check_network_element_on_site_by_site_id() – проверка сетевых элементов на позиции
- check_alarms_in_incidents_of_group_for_reassignment() – проверка аварии в инциденте для определения группы для переназначения;
- check_network_element_on_site_by_site_id() – проверка наличия сетевых элементов на позиции;
- check_incident_at_reclamation_for_reassigment – проверка признаков рекламации в инциденте;
- check_incident_by_work_time_bot_for_reassigment() – проверка инцидента по времени взятия в работу ботом.

Собственно, сами модули main_dispatcher_fm_bot_1_1.py и

main_dispatcher_fm_bot_1_2.py содержат 3 основных функции для работы:

- telegram_request() – для отправки запросов боту, который уведомляет разработчиков о критических сбоях.
- run_http_request() – функция для отправки запроса к RESP API NSN портала и собственно всей логикой обработки ответа.
- update_token() – функция для обновления токена для доступа к NSN порталу.
- run_schedule() – главная функция в работе ботов, создающая расписание выполнения остальных функций. Например, обновление токена или сохранение и архивация логов раз в неделю или запуск обработки инцидентов по времени, указанном в конфигурации бота.

Изначально планировали создать базу данных на PostgreSQL для хранения обрабатываемых инцидентов, в последствии отказались от этой идеи так как у нас уже есть база данных, и мы получаем данные об инцидентах с помощью API запросов. База данных осталась только для хранения токенов ботов.

Разработанные сервисы не имеют интерфейса и наблюдать за их работой можно, проверив статус systemd сервиса и читая лог файл. В лог файлах фиксируется вся информация об обработанных инцидентах. А чтобы они не занимали слишком много памяти они регулярно архивируются средствами разработанного модуля (рис. 6).

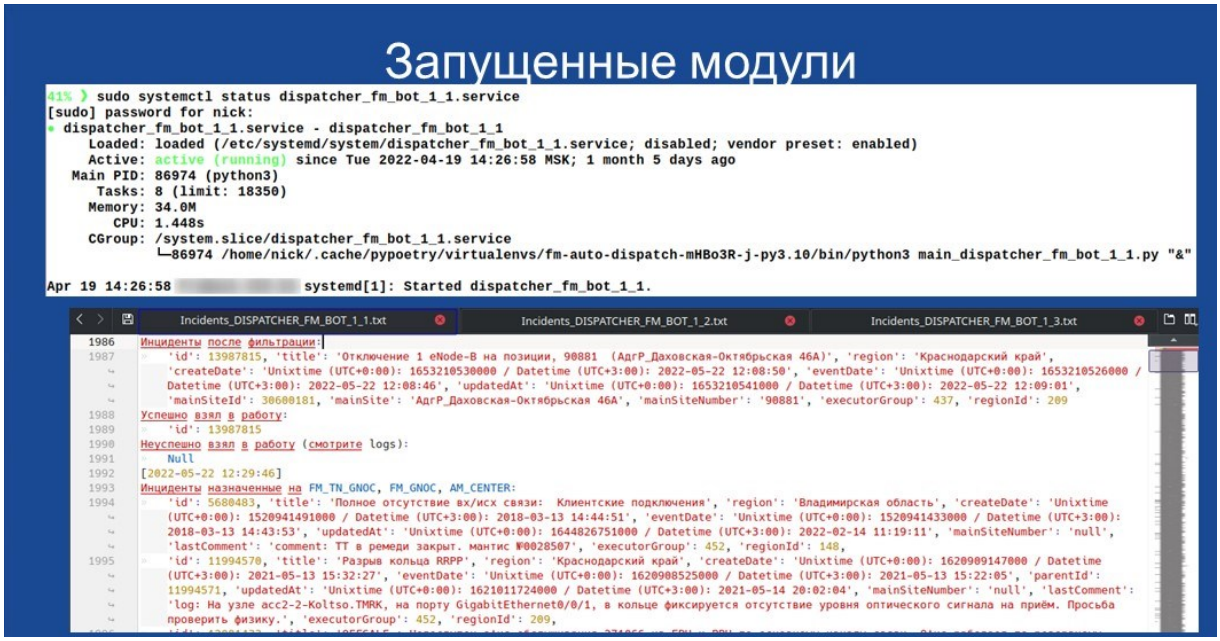

Рисунок 6. Статус systemd сервиса и лог файл первого бота

Следствия работы ботов можно наблюдать в логах системы IMS. Например, инцидент из г. Махачкала – на позиции обнаружена авария по перегреву, инцидент назначен на местную эксплуатацию с необходимым комментарием (рис. 7).

Еще один пример, инцидент из г. Тверь, до базовой станции отсутствует канал управления (рис. 8).

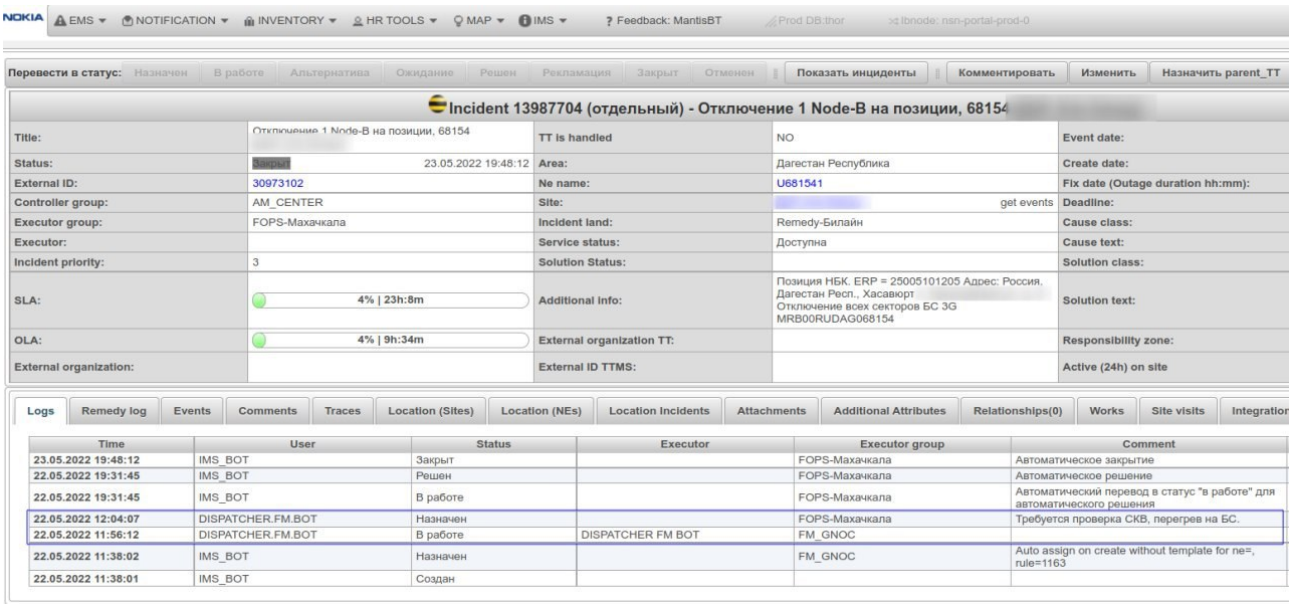

Рисунок 7. Инциденты в системе IMS: обработанный инцидент г. Махачкала [4]

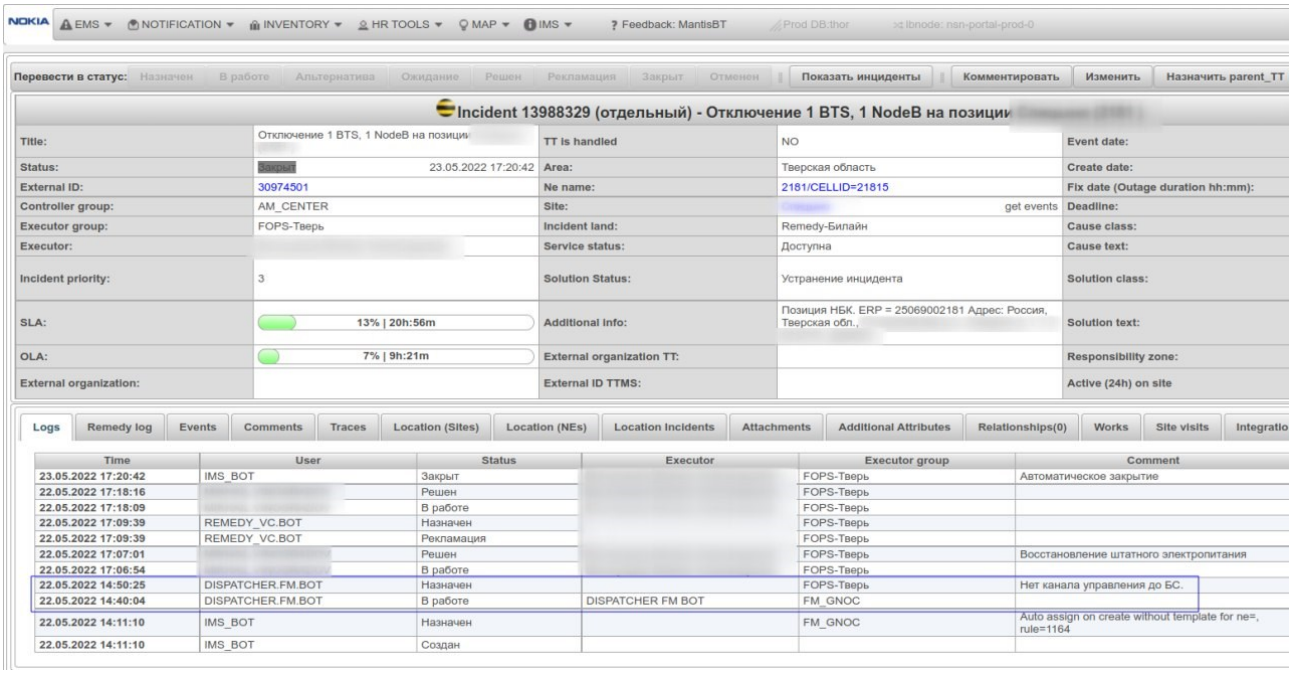

Рисунок 8. Инциденты в системе IMS: обработанный инцидент г. Тверь [4]

Заключение

В результате анализа процесса обработки инцидентов, сделан вывод о необходимости разработки модуля автоматизации – разработаны алгоритмы обработки и диспетчеризации инцидентов, разработан, протестирован и развёрнут на сервере организации модуль автоматизации обработки инцидентов в Service Desk системе IMS.

На этапе внедрения разработанный модуль обрабатывает инцидентов больше, рядового инженера отдела мониторинга за смену, что высвобождает отделу приблизительно 15-20 человеко-часов в сутки.

Для дальнейшего расширения функционала, в рамках интеграции разработки в проект DigiMOP на развернутый KNIX сервер (open source разработка инженеров глобальной Nokia) принято решение о переносе функции данного модуля в виде отдельных микросервисов.

СПИСОК ИСТОЧНИКОВ

1. https://rkn.gov.ru/news/rsoc/news67782.htm – Роскомнадзор. «Большая четверка» сотовых операторов продолжает наращивать темпы развития сетей мобильной связи.
2. https://rspectr.com/articles/operatory-prosyat-chastot – операторы просят частот.
3. https://www.gsmvezde.ru/articles/chto_nujno_delat_esli_telefon_ploho_lovit_set – что делать, если телефон не ловит сеть мегафон, билайн, мтс.
4. https://portal.nsn-net.ru/nsn-portal/ims/one.js – портал NSN, скрины обработанных инцидентов.

DEVELOPMENT OF AN INCIDENT PROCESSING AUTOMATION MODULE FOR SERVICE DESK SYSTEMS

© 2022 N. V. Baranov, A. N. Zelenina

Voronezh Institute of High Technologies (Voronezh, Russia)

The development of an incident handling automation module for Service Desk systems is presented. It consists of several backend services (bots) running on a Linux server. The planned scope is a provider or an outsourcing company that handles incidents in Service Desk systems. The advantage the proposed automation solution, in comparison with analogues, is that services (bots) automatically check the relevance of tasks, process typical routine tasks, compile and submit a report to the incident, and appoint responsible persons. This will save monitoring engineers and the troubleshooting department from repeating routine tasks many times when handling typical incidents.

Keywords: incident handling in Service Desk systems, service (bot).