Удалённый мониторинг системы

# Задачи:

1. Удалённо просматривать загруженность метрик систем.
2. Получения уведомлений об ошибках.

# Вступление:

Многие предприятия приобретают большое количество ПК и нанимают к себе на работу системного администратора (СА), в обязанности которого входим контроль работы всех ПК предприятия. На предприятии многие ПК работают по 24 часа в сутки, СА не может следить за данными ПК всё время, сбой в работе может произойти и в ночное время, что очень негативно повлияет на работу системы, а может и всего предприятия. Многие проблемы можно решить ещё до их появления, в данном случаи необходимо ПО которое будет производить мониторинг системы круглосуточно и сообщать о превышении значения какой-либо метрики снятой с определённого ПК.

# Используемые технологии:

ПО реализовано на языке программирования Java с использованием Spring MVC, Hibernate Framework и highcharts. Для работы ПО необходима БД (строгого ограничения в СУБД нету), которая будет хранить всю информацию. Присутствует возможность использования двух разных БД, одна из которых будет хранить всю необходимую информацию о подключения к удалённым ПК, другая будет хранить большое количество данных снятых в процессе работы.

# Краткое писание:

System Monitoring (SM) – это приложение, позволяющее непрерывно отслеживать состояние хоста, снимать значения его метрик, предоставлять это все в удобном для пользователя виде и оповещать пользователя о проблемах связанных с доступностью хостов или со значениями метрик. UI (Пользовательский интерфейс) реализован в виде веб сайта, это удобно тем, что пользователь в любой момент из любого места (где есть выход в интернет) сможет использовать данный сервис.

В SM, функции доступные пользователю, изначально доступны в зависимости от одной из двух ролей: пользователь или администратор.

Возможности, предоставленные пользователю:

• Добавление нового хоста

• Просмотр сведений о хосте

• Просмотр возникших проблем

• Система оповещения

• Графическое представление для состояния метрик

• Настройка графиков для главной страницы

Администратору предоставлены все возможности пользователя, а также следующие функции:

• Редактирование хостов

• Редактирование локальных метрик

• Редактирование шаблонных метрик

• Редактирование сервисов для хоста

• Добавление новой шаблонной метрики

• Редактирование аккаунтов

# Архитектура SM:

Архитектуру можно разбить на несколько частей:

1. UI – интерфейс пользователя.
2. Controllers – слой который обеспечивает связь между пользователем и системой.
3. Agents – Агенты управляющие жизненным циклом ПО.
4. DB – Слой базы данных, для хранения необходимой информации.

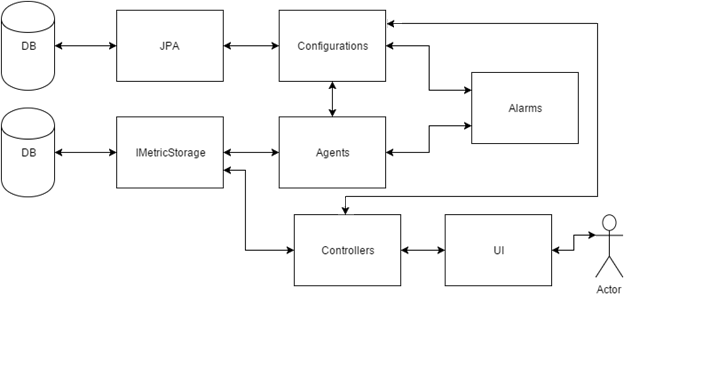


Рис.1

Слой BD состоит из двух баз данных, с одной из которых взаимодействует реализация интерфейса IStorageController, она хранит в себе информацию о метриках и их значениях. Взаимодействие со второй БД происходит через интерфейс JPA, в этой базе хранятся информация о хостах и уведомлениях.

Сервис “SchedulerTask” запускает по расписанию Агентов, которые отвечают за подключение к хосту, считывании значений метрик с хостов и за отправку уведомлений.

Уведомления в системе делятся на два типа:

1. Отправка сообщений на почту.
2. Отправка уведомления на UI пользователя.

Контроллеры обеспечивают связь между пользователем и системой: контролируют ввод данных пользователем и используют модель и представление для реализации необходимой реакции.

Слой UI представлен в виде Web интерфейса, который использует технологию Thymeleaf для построения страниц.

# Детальный дизайн ядра системы:

Центральной частью ПО является Spring Framework, который отвечает за управление жизненным циклом объектов: создание объектов, вызов методов инициализации и конфигурирование объектов путем связывания их между собой. Конфигурирование контейнера осуществляется путем аннотаций и класса DatabaseConfig, содержащих информацию, необходимую для создания bean’ов.

Для работы с базой данных предусмотрен интерфейс IStorageController, в который обращается контроллер за необходимыми данными. Реализация IStorageController в свою очередь обращается в одну из реализаций отвечающую за определённую сущность и возвращает необходимые данные в контроллер.

После запуска приложения Spring разворачивает Web сервер и инициализирует beans, далее с помощью аннотации Scheduler, каждые 10 секунд сервис “ScheduledTask” запускает считывание метрик с удалённых хостов.

Сервис “ScheduledTask” поочерёдно считывает информацию о каждом хосте, которая хранится в базе, создаёт объект класса SSHAgent, который по сетевому протоколу SSH подключается к удалённому хосту , если подключение не удалось установить, то с помощью сервиса “RouteAlarms” происходит отправка сообщения об ошибке , если подключение удалось установить, то интерфейс IStorageController возвращает все InstanceMetric для данного хоста и для каждой метрики SSHAgent создаёт новый канал, отправляет команду считавшую из поля в InstanceMetric и возвращает число (значения метрики в данный момент) . После получения значения метрики сервис “ScheduledTask” сравнивается полученное число с пороговыми значениями записанными в InstanceMetric и если значение метрики выходит за порог, то с помощью сервиса “RouteAlarms” происходит отправка сообщения об ошибке. Далее цикл для сервиса “ScheduledTask” повторяется.

В системе предусмотрена защита ввода в адресную строку адреса недоступного для данного пользователи или “Инжекции” в файл javascript функции, которая может получить данные доступные только администратору. При попытки доступа к данным обычным пользователем, он будет перенаправлен на страницу с кодом ошибки 404, хотя данная ошибка не отражает ошибку связанную с правами доступа, но такой подход был выбран для того, что бы пользователь не мог точно знать существует ли данная функция на сервере.

# Вывод:

Из всего вышесказанного можно сделать вывод, что данное приложение решает поставленные задачи. Данный сервис позволит вовремя получать необходимую информацию для того, чтобы принять нужное решение и устранить проблему.