Website Monitoring Tool

(Тестовое задание на позицию Java-девелопера)

Нужно написать приложение, которое будет мониторить состояние
одного или нескольких внешних веб-приложений путем периодического выкачивания
страниц по заданным URL-ам, и контроля времени ответа серверов, HTTP кода респонса,
размера респонса в некотором заданном диапазоне, и дополнительно (опционально) -
наличие некоторой подстроки в респонсе.

Подобную функциональность включают в себя существующие системы мониторинга серверов,
например, Nagios, Zabbix.

Сервер

Серверная часть приложения должна периодически опрашивать все заданные в конфигурации
приложения URL-ы, и отдавать их статус в пользовательский интерфейс.

Пользовательский интерфейс

Основной пользовательский интерфейс приложения должен состоять из одной HTML страницы,
которая показывает статус всех URL-ов в виде таблицы.

Таблица должна периодически перезагружаться с сервера,
чтобы отображать актуальный статус.

Статус каждого URL-а может принимать три состояния: OK, WARNING, CRITICAL.

Кроме статуса, в таблицу нужно выводить другие полезные детали о результатах мониторинга URL-ов,
чтобы было легче понять, в чем состоит проблема.

При статусах WARNING и CRITICAL, страница должна издавать звуковой сигнал (разный для WARNING и CRITICAL)

Конфигурирация приложения

Конфигурация URL-ов, которые подлежат мониторингу приложением, должна задаваться в базе данных.

Для каждого URL-а, который будет мониториться приложением, задаются такие параметры:

* Период мониторинга. Должна быть возможность задать период различным для разных URL-лов.

* Время ответа сервера (отдельные пороги для OK, WARNING, CRITICAL)
  В случае задержки с ответом какого-то URL-а, остальные проверки не должны задерживаться,
  и их результат должен отображаться пользователю.

* Ожидаемый HTTP response code.
  Если сервер возвращает какой-то другой код, статус URL-а должен быть CRITICAL.

* Ожидаемый диапазон размера респонса в байтах (min и max).
  Если размер контента выходит за пределы допустимого диапазона, статус URL-а
  должен быть CRITICAL.

* Опционально - подстрока, которая должна содержаться в респонсе.
  Если подстроки нету, статус URL-а должен быть CRITICAL.

Тесты

В процессе девелопмента, тестируйте отдельные модули с помощью junit- или "ручных"
тестов. При тестировании какого-то модуля, другие модули, от которых зависит этот модуль,
можно заменять заглушками, упрощенно реализующими интерфейсы этих модулей.

После завершения девелопмента тесты не удаляйте, оставьте их в коде.

Детали реализации

Приложение нужно реализовать в основном на JDK/J2EE, по возможности с минимальным использованием
сторонних библиотек.

Приложение должно быть собрано в виде war файла.
Предоставьте скрипт компиляции и сборки приложения с помощью любого билд-тула,
с которым Вы знакомы.

Используйте любой сервлет-контейнер, с которым Вы знакомы.

Несмотря на простоту задачи, приложение следует реализовать максимально объектно -

в виде нескольких модулей, связанных через тонкие интерфейсы. Так, чтобы это приложение
можно было легко расширять и наращивать функциональность в будущем, и чтобы мы могли
оценить ваши навыки в области объектного дизайна.

Примеры возможного расширения функциональности:

* Использование другого пользовательского интерфейса
  (например, native приложения, возможно, через REST Web-API)

* Другой способ хранения конфигурации мониторинга, например, конфигурационный файл вместо базы данных.

* Сохранение результатов мониторинга в базу данных, формирование отчетов за интервалы времени в прошлом
  (периоды проблем, даунтайм)

* Альтернативные способы мониторинга URL-ов, например мониторинг через удаленные прокси-агенты,
  чтобы проверять доступность интересующих нас URL-ов из разных мест в Интернете.

* Временное исключение URL-а из мониторинга. Например, когда оператор увидел проблему с каким-то URL-ом,
  и уже занимается ее решением, в этом случае можно приостановить сигналы мониторинга.
  (в Nagios эта фича называется Acknowledge service problem)

* Scheduled downtime - приостановить мониторинг, если мы сами останавливаем
  сервис для обслуживания (похоже на предыдущий пункт).

Будет плюсом, если Вы заложите какие-то из этих расширений в структуру приложения
(в виде интерфейсов без полноценной реализании, с реализацией в виде заглушек).

Пожалуйста, представьте общую UML диаграмму приложения, на которой были бы видны модули,
и взаимодействие между ними. По возможности, предоставьте UML диаграммы модулей.

Помимо просто работоспособности приложения, мы придаем большое значение аккуратности, стилю кода,
и хорошему объектному дизайну.

Уровни реализации:
==========

Junior
------

* Клиент-серверное приложение с веб-интерфейсом, тестирующее несколько сайтов с сервера параллельно,
  и выводящее результат в браузер.

  Результат мониторинга выдается в виде таблицы на HTML странице (через сервлет или jsp).

  Временное выключение/включение мониторинга URL-а через веб-интерфейс.

  Конфигурация мониторинга (добавление, настройка, удаление URL-ов мониторинга)
  задается через веб-интерфейс и сохраняется в базе данных.

  Используйте любой web-application framework, и любую базу данных, с которыми Вы знакомы.

Middle / Senior
---

* То же, что и выше, плюс работа через прокси-агенты на удаленных машинах (аналогично nrpe в Nagios).

  Это нужно для того, чтобы проверить доступность сайта из разных точек в интернете.

  Центральный сервер, где установлено основное приложение, раздает задачи приложениям-агентам,
  установленным на нескольких удаленных серверах в разных точках интернета, и собирает
  с них результаты мониторинга URL-ов.

  Результатом выполнения этого уровня будет два веб-приложения - основное приложение
  и удаленный агент. Удаленный прокси-агент также может быть консольным приложением, или веб-приложением.

* Реализуйте структуру приложения согласно возможным расширениям функциональности, описанным выше,
  а также вашими идеями, как можно было бы расширить/улучшить систему.

* Предоставьте качественные UML диаграммы, отражающие основные компоненты системы и суть взаимодействия между ними.

Оформление результатов
==========

Этот пункт очень важен, обратите, пожалуйста, внимание.

Когда Вы сообщите, что приложение готово, мы предоставим вам вход на виртуальную Linux-машину
(IP адрес, логин/пароль) для инасталляции приложения, чтобы приложение было доступно в Интернете.

Если у Вас есть свой тестовый сервер с уже настроенной инфраструктурой для запуска веб-приложений,
можно использовать его.

Если Вы реализуете мониторинг через прокси-агенты, мы дадим вам несколько дополнительных
виртуальных Linux-машин для установки агентов.

Туда вам нужно будет установить сервлет контейнер, ваши приложения, и дать нам URL, куда можно
зайти, чтобы посмотреть результат.

Выберите несколько URL-ов мониторинга для примера, добавьте в конфигурацию
приложения, чтобы были какие-то данные для просмотра.

Результаты нужно представить в одном полном пакете, который должен включать:

1. Текстовый README файл, описывающий кратко следующее:

    - какие части функциональности Вы реализовали

    - URL-ы на тестовом сервере, где можно быстро посмотреть результат

    - краткое описание того, как работает система, интерфейс, какие есть особенности, и т.п.

    - какие использовались технологии для различных частей системы

    - на какой операционной системе велась разработка, в каком IDE,
      какой веб-сервер и билд-тул использовался

    - инструкции по сборке приложения, рассчитанные на человека, не знакомого с вашим билд-тулом,
      тоесть полную последовательность консольных команд для установки нужного окружения и для сборки.

2. Полный исходный код приложения. Не нужно заливать его на публичные репозитории кода,
   такие как github, bitbucket, и т.п.

3. UML диаграммы:

    - компонентную диаграмму всего приложения, где было бы видно,
      из каких компонентов оно состоит, и как они взаимодействуют

    - по необходимости, диаграммы отдельных компонентов

4. war файл приложения, который можно легко задеплоить на веб-сервер,
   и увидеть результат локально

Соберите, пожалуйста, все это в один zip-архив, и пришлите одним письмом все сразу.

Коммуникация в процессе работы над заданием

Дайте нам знать, пожалуйста, когда Вы беретесь за выполнение задания, и в дальнейшем,
сообщайте раз в несколько дней, как продвигаются дела. Если по каким-то причинам
Вы прекратите работу над заданием, сообщите об этом тоже.

Мы постарались включить в текст задания все необходимые детали,
но если что-то не совсем ясно, можно созвониться с техническим специалистом нашей
компании, и все уточнить.

Что делать, если у вас возникли вопросы по заданию

1. Перечитайте текст задания еще раз внимательно, т.к. ответы на большинство часто задаваемых вопросов
   уже содержатся в задании.

2. Некоторые моменты в задании опущены, чтобы дать вам возможность самостоятельно решить, как лучше сделать.
   Это типичная ситуация при девелопменте - в постановках реальных задач обычно нету полной определенности.

3. Если вопросы остаются, обратитесь к представителю нашей компании, наш технический специалист обсудит это с вами.

--

Спасибо за Ваш интерес к этой вакансии.