Данные осмотра из базы данных выводятся в удобном, читаемом для врача виде (рис. 5).

Практический результат работы: разработано Android-приложение с упрощенным интерфейсом для быстрого осмотра пациентов с возможностью сохранения результатов осмотра. Целевая аудитория – врачи-клиницисты поликлиники, ведущие амбулаторный прием. Разработанное приложение позволит оптимизировать работу врачей в медицинском центре материнства и детства, путём сокращения времени на заполнение данных об осмотре, что увеличит возможность для непосредственного общения между врачом и пациентом.

Рис. 5. Вид отображения вопросов и ответов осмотра

1. Build anything on Android [Электронный ресурс] – URL: https://developer.android.com/
2. Android Layout xml [Электронный ресурс] – URL: https://coderoad.ru/
3. MySQL [Электронный ресурс] – URL: https://metanit.com/sql/mysql/2.1.php

**Рубрика: : Информационные технологии: теория и практика**

УДК 004.453

# ПОДХОДЫ К СОЗДАНИЮ СИСТЕМЫ УПРАВЛЕНИЯ ПАРСЕРАМИ ПРИ РАЗРАБОТКЕ АГРЕГАТОРА

**Е.А. Бова, М.В. Водяницкий, Д.А. Мальцев, А.Д. Шнейдер**
бакалавры
**О.Б. Богданова**
преподаватель

*Владивостокский государственный университет экономики и сервиса*
*Владивосток. Россия*

*Данная статья описывает структуру системы управления парсером, взаимодействие подсистем и примеры решения проблем, возникающих при создании таких систем в процессе разработки агрегатора.*

***Ключевые слова:*** *парсер, агрегатор, система управления, логирование, распределение заданий, мониторинг.*

# PRINCIPLES OF CREATION PARSER MANAGEMENT SYSTEM IN DEVELOPING AN AGGREGATOR

*This article describes parser management system structure, relations between elements and how to solve difficult to create this in developing an aggregator.*

***Keywords:*** *parser, aggregator, management system, logging, task management, monitoring*

**Введение**

Агрегаторам контента нужно получать информацию с большого количества ресурсов. Одного парсера, как правило, для этого недостаточно, что вызывает необходимость в большом количестве парсеров, однако это приводит к проблемам управления, распределения задач и мониторинга состояния их работы. Именно эти задачи решает система управления парсерами. Есть множество подходов к созданию такой системы, которые будут рассмотрены в данной статье.

Такая система функционально состоит из следующих подсистем:

– Системы запуска,

– Системы распределения задач,

– Системы мониторинга ошибок,

– Системы мониторинга работы парсеров.

В качестве примера будет использоваться агрегатор контента, который собирает информацию из разных магазинов игр.

**Система запуска**

В основном парсеры запускаются на Linux машинах для удобной настройки окружения, параметров парсера и требуемых инструментов или пакетов. Парсеры можно запускать и чистом Linux, но для нормальной работы это потребует установки дополнительных пакетов определённых версий, что не всегда возможно.

Для решения подобных проблем практикуется применение изоляции или паравиртуализации, например, Docker, в котором формируется и настраивается при билде окружение. Такие контейнеры просты для разворачивания на любой машине – достаточно наличие VPS машин и запущенных на них Docker контейнеров для дальнейшей регистрации парсеров и запуска их основного скрипта. Для разворачивания контейнеров могут также использоваться аналоги Docker – Podman и Builah [7].

Применяемая контейнеризация упрощает процесс обновления парсеров и перезапуска контейнера, в частности, при возникновении критических ошибок.

В случае, когда для работы агрегатора используется множество разных парсеров, запускаемых в разных контейнерах, применяют оркестрацию контейнеров [5], а именно такие продукты как Kubernets и Docker Swarm.

**Система мониторинга работы парсера**

Логика работы мониторинга парсера прописывается в самом парсере – программа должна в соответствии с конфигурацией сервера сообщать ему свой статус, перечень которых определен заранее, и запрашивать конфигурацию у сервера (рис. 1). Получение запросов позволяет иметь серверу актуальное состояние парсера. Отсутствие ответов от парсера идентифицируется по отсутствующим обновлениям. При получении запроса от парсера сервер логирует его состояние, в том числе время его прихода. Специальная задача периодически проверяет и сравнивает последнее время соединения. При превышении заданного временного порога запущенный парсер помечается как неактивным, и сервер уведомляет администратора об этом.

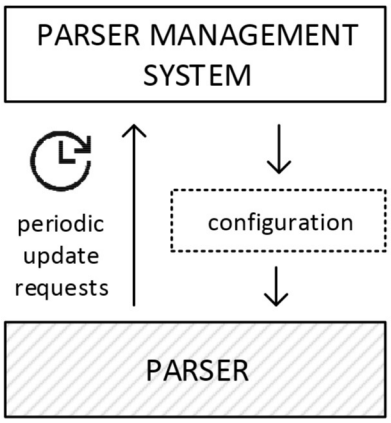


Рис. 1. Схема работы системы мониторинга

Для непрерывности выполнения заданий невыполненная последняя работа парсера снова помещается в пул задач через определенное время для избегания таких ситу-

аций, как остановка работы парсера при нестабильном коннекте с сайтом. Для связи с сервером активно используется концепция RESTful API. Существуют еще другие способы, такие как HTTP keep-alive и TCP сокет, но они требуют усложнения архитектуры и менее надежны.

**Система мониторинга ошибок**

Сервер в конфигурации задает правило сбора логов: сбор всех логов, сбор логов с ошибками, полное отключения сбора логов – только для случая, когда имеются проблемы на стороне системы управления.

Лог пишется постоянно и сохраняется в текстовых файлах локально вне контейнеров. Лог файлы в зависимости от конфигурации делятся на файлы по времени. Ошибки, возникающие при работе парсера, можно разделить на 2 группы (табл. 1).

*Таблица 1*

**Возникающие ошибки**

| Показатели | Обрабатываемые ошибки | Необрабатываемые ошибки |
|---|---|---|
| Общее описание | Предусмотрены при работе парсера, тип ошибки заранее прописан и хорошо идентифицируется в системе управления | Связаны с неожиданным завершением работы парсера, как правило, невозможно идентифицировать автоматически, обязательно скидывается полный последний лог, с которым работают разработчики |
| Пример возникновения | Ошибка в разметки страницы при парсинге, отсутствие коннекта с целью парсинга | Ошибки, вызванные инструментарием, поломка окружения, ошибки чтения конфигураций, ошибки на стороне хостинга |

В зависимости от архитектуры парсера возможны два пути реализация системы сбора. В случае использования архитектуры микросервисов, как правило, реализуется стек ELK [6], состоящий из Elasticsearch – движка для хранения и поиска данных, Logstash – утилиты для индексирования логов, Kibana – веб-сервиса для просмотра логов. В некоторых реализациях вместо Logstash используется Fluentd [4], так как этот программный комплекс имеет больше возможностей передачи логов для хранения и менее требователен к ресурсам, чем Logstash [1]. Иногда используется syslog-ng в качестве сборщика логов, где сильно ограничены ресурсы, но необходимо выполнять потенциально сложную обработку.

При реализации парсера как монолитного приложения (рис. 2), которое будет запускаться в виде множества экземпляров, можно использовать более простые решения, такие как сохранение логов в виде файлов, в том числе и в системе управления. В БД отправляется запись о том, что за лог, от кого, когда, какой тип парсера, его последняя конфигурация, его версия и прочее. В случае возникновения обрабатываемой ошибки парсер должен сообщить о ней, в частности, тип ошибки и место возникновения, на сервер через специальный API.

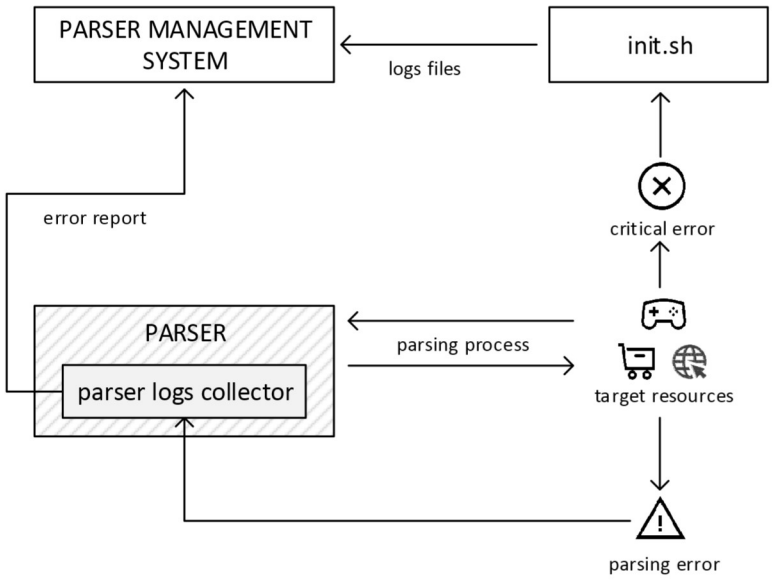


Рис. 2. Схема работы системы мониторинга ошибок для монолитного приложения

В крупных проектах может использоваться Sentry – более продвинутый инструмент для мониторинга ошибок. Данная утилита может работать с широким списком языков программирования и является OpenSource проектом, но требует много ресурсов для своей работы. На рынке имеются аналоги Sentry, такие как Rollbar, Raygun, Airbrake, Bugsnag, Logentries [2] и другие.

**Система распределения задач**

Запросы на задания создаются специальным формирователем заданий, запускающимся с определенным интервалом, так как данная задача является ресурсоемкой. Функция формирователя – создать записи заданий, например: собрать данные с определенной страницы (скрины, описание и прочее), получить обновление цен с определенных страниц.

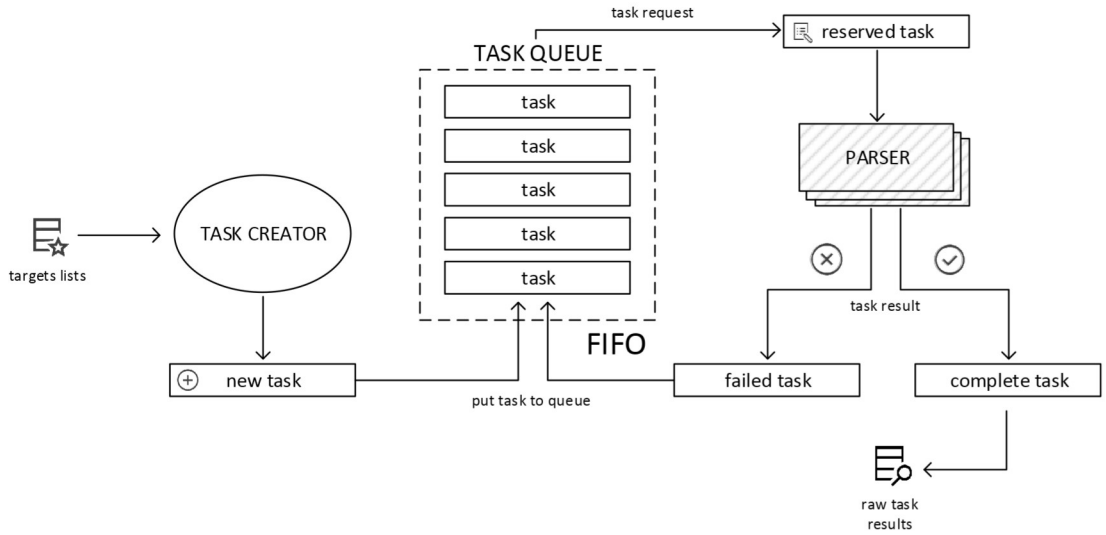


Рис. 3. Схема работы системы распределения задач

Задания попадают в общий пул и по принципу FIFO раздаются при запросе заданий со стороны парсеров (рис. 3). В качестве планировщика выполнения заданий может использоваться такие утилиты как cron, Anacron, fcron, Hcron и Jobber [3]. Сравнение планировщиков приведено в табл. 2.

*Таблица 2*

**Сравнение планировщиков заданий**

| Показатели | Достоинства | Недостатки |
|---|---|---|
| Cron | Простое, надежное и распространённое решение, которое часто включается в другие системы | Отсутствие гибкости в возможностях настройки |
| Anacron | Возможность автоматического выполнения после включения в том случае, если в запланированное время задание система была выключена | Возможности планирования сильно ограничены, нельзя настроить на поминутное / почасовое выполнение |
| Hcron | Возможность группировки заданий, управление сетями систем, создание cron заданий для задач, использующих системы контроля версий | Не обновляется (последнее обновление – в 2010 году) |
| Jobber | Широкие возможности для гибкой настройки заданий, наличие уведомлений о неудачных запусках, работа с проваленными заданиями | Избыточен для систем с уже встроенными планировщиками, например Laravel |

Задачи типизированы, например, поиск игр, добавление новой игры, обновление цены, обновление игры. Это необходимо для дальнейшего работы, так как после окончания задания следуют разные действия.

Все данные с парсеров попадают в специальные таблицы «сырых данных». С полученными данными работают другие обработчики через определенный интервал, например после поиска игр нужно выделить новые и занести их в таблицу новых обнаруженных игр, позже формирователь заданий создаст новую задачу – найти новую игру, получить описание и так далее.

Помимо этого есть глобальная программа-модератор всех временных записей и заданий. Она проверяет дату создания и обновления записи, в случае устаревания – удаляет лишние, логирует это.

**Заключение**

Реализация каждого компонента системы управления парсерами может быть как разработана самостоятельно, так и внедрена как готовое решение. Выбор конкретного продукта или реализация метода зависит масштабов проекта, его бюджета и опыта разработчиков.


1. 5 Awesome Logstash Alternatives [Электронный ресурс]. – URL: https://sematext.com/blog/logstash-alternatives/
2. 20 best alternatives to Sentry as of 2021 [Электронный ресурс]. – URL: https://www.slant.co/options/964/alternatives/~sentry-alternatives
3. Альтернативы Планировщика Задач cron [Электронный ресурс]. – URL: https://sites.google.com/site/610393/blog-1/nastrojka/cron/planirovsik-zadac
4. Как настроить централизованное логирование для Docker Swarm с помощью Fluentd [Электронный ресурс]. – URL: https://mcs.mail.ru/help/ru_RU/cases-logs/case-swarm
5. Оркестрация контейнеров [Электронный ресурс]. – URL: https://www.xelent.ru/blog/chto-takoe-orkestratsiya-konteynerov/
6. Установка и настройка ELK, сбор, анализ и визуализация логов [Электронный ресурс]. – URL: https://mcs.mail.ru/help/ru_RU/cases-logs/case-logging
7. Podman и Buildah для пользователей Docker [Электронный ресурс]. – URL: https://habr.com/ru/company/redhatrussia/blog/467105/



**Рубрика: Информатизация на предприятиях**

УДК 004.415.2


# РАЗРАБОТКА СЕРВИСА ДЛЯ СОЗДАНИЯ ОТЧЁТОВ КОМПАНИИ «ООО ТРАСТ НЕДВИЖИМОСТИ» г. ВЛАДИВОСТОК


**Я.А. Бондаренко**
бакалавр
**Е.В. Кийкова**
преподаватель

*Владивостокский государственный университет экономики и сервиса.*
*Владивосток. Россия*



*Формирование отчётности играет большую роль в деятельности любой компании. Многие компании до сих пор формируют большое количество различных отчётов вручную, из-за чего возникают ошибки и затя-гиваются сроки сдачи. В статье рассматривается вопрос разработки сервиса для автоматизации бизнес-процессов связанных с формированием отчётности в виде сервиса с использованием фреймворков Vue и Lara-vel на примере компании «ООО Траст недвижимости».*



***Ключевые слова:*** *отчётность, автоматизация, сервис, vue, laravel..*


# DEVELOPMENT OF A SERVICE FOR CREATING REPORTS FOR THE COMPANY "OOO TRAST NEDVIZHIMOSTI", VLADIVOSTOK


*Reporting plays an important role in the activities of any company. Many companies still compile a large number of different reports by hand, which leads to errors and delayed deadlines. The article discusses the issue of de-veloping a service for automating business processes related to the formation of reporting in the form of a service using the Vue and Laravel frameworks on the example of the company «OOO Trast nedvizhimosti»*



***Keywords****: reporting, automation, service, vue, laravel.*