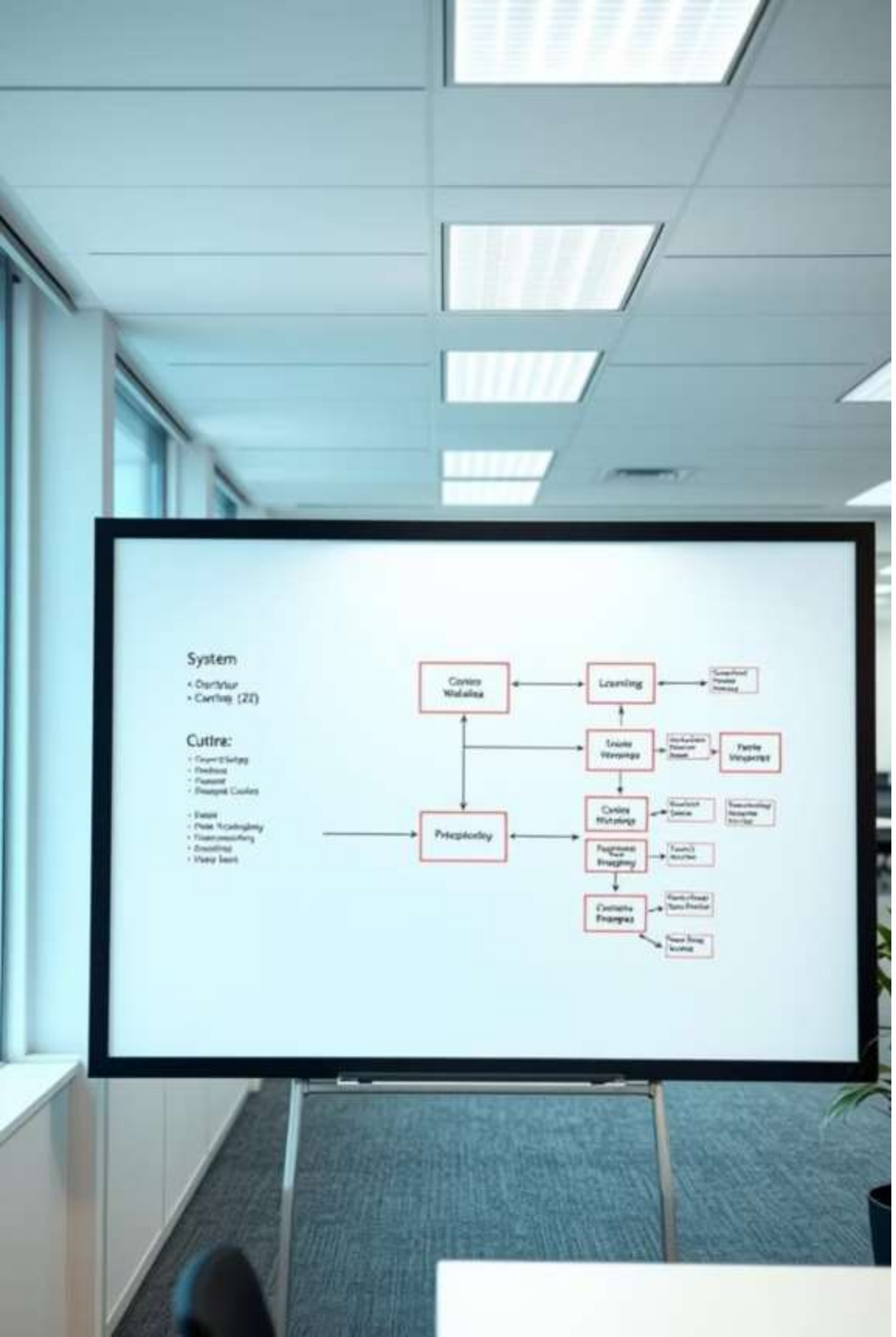
Научно-исследовательская работа

Тема: Инструмент для отслеживания обновлений на интернет-ресурсах

Автор: Куртяков Александр Алексеевич, С21-501

Научный руководитель: Овчаренко Евгений Сергеевич



Введение: Актуальность и цели НИР

В условиях быстрого развития цифровых платформ и постоянного увеличения объемов информации, задача оперативного отслеживания изменений на веб-ресурсах становится крайне важной. Ручной мониторинг многочисленных источников, таких как *GitHub, Stack Overflow, Habr, YouTube,* занимает часы и часто оказывается неэффективным.

Цель исследования — сократить время проверки актуальности информации за счет централизованных уведомлений в *Telegram.* В текущем семестре основное внимание уделено расширению функциональности и повышению надежности системы.

Ключевые доработки системы

1

Интеграция с Habr и YouTube

Расширен спектр отслеживаемых источников за счет подключения к популярным платформам *Habr* и *YouTube Data API*.

2

Валидация пользовательских ссылок

Добавлены механизмы проверки доступности, корректности формата и поддерживаемых доменов для ссылок.

3

Оптимизация обработки ошибок

Реализованы улучшенные механизмы логирования и защиты от отказов *API*, повышающие стабильность системы.

4

Автоматическое тестирование

Внедрены юнит-тесты для основных модулей, включая новые интеграции и валидацию ссылок.

5

Масштабируемая архитектура

Улучшена архитектура обработки ссылок для упрощения добавления новых ресурсов в будущем без переработки ядра.

Результаты НИР и эффективность

В результате исследования и разработки достигнуты следующие ключевые результаты:

- Система поддерживает *4* платформы: *GitHub, Stack Overflow, Habr, YouTube.*
- Корректное добавление и валидация ссылок через *Telegram*-бота.
- Интеграции с *Habr* и *YouTube* работают стабильно.
- Код покрыт автоматическими тестами для новых функций.
- Масштабируемая архитектура, готовая к расширению источников.

Расчет эффективности

До внедрения системы пользователи тратили *3-5* часов в неделю на ручной мониторинг. После внедрения, время на взаимодействие с системой не превышает *5-10* минут в неделю. Система сокращает время отслеживания информации в *30-40* раз, что полностью соответствует поставленной цели исследования.

Анализ предметной области: Проблемы и решение

Отсутствие единого интерфейса

Пользователям приходится вручную проверять множество ресурсов, а встроенные подписки часто ограничены и неудобны, что приводит к пропуску важной информации.

Разнородность уведомлений

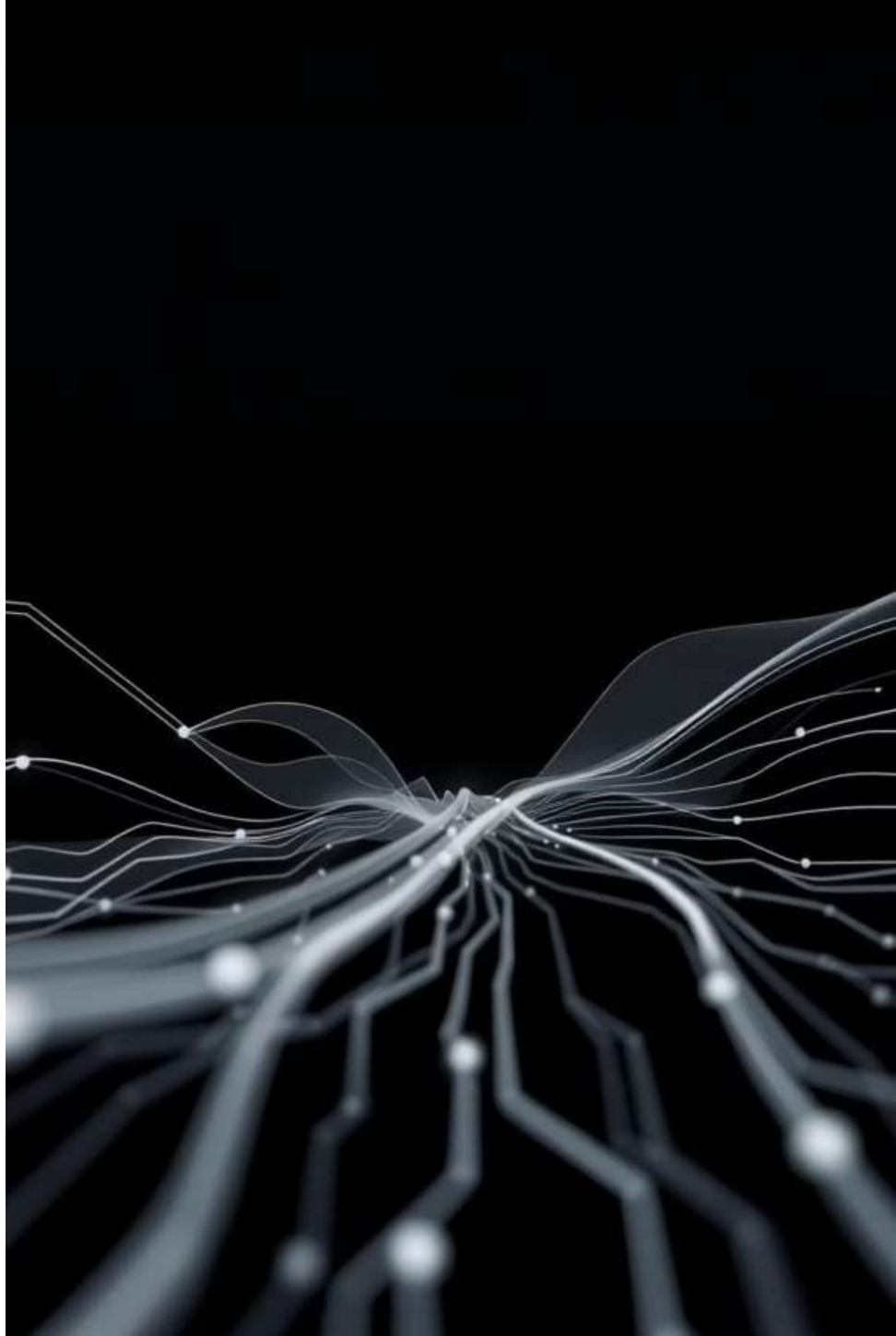
Различные платформы *(GitHub, Stack Overflow, Habr, YouTube)* используют разные форматы уведомлений, снижая удобство и эффективность мониторинга.

Недостатки существующих решений

Существующие решения *(Visualping, Distill.io)* ограничены веб-скринингом, не интегрируются с популярными *API* и не предлагают централизованного интерфейса через чат-боты.

Telegram как платформа

Telegram предлагает богатый *Bot API*, быструю доставку сообщений, поддержку *UI*-элементов и кроссплатформенность, что делает его идеальной платформой для централизованных уведомлений.



Обзор отслеживаемых ресурсов

В текущем семестре система расширена за счет интеграции с *Habr* и *YouTube*.

Платформа	Особенности	Метод интеграции
Habr	Крупнейшая русскоязычная ИТ-платформа, включающая статьи, блоги, обсуждения и комментарии.	Парсинг *RSS*-лент
YouTube	Крупнейшая видеоплатформа в мире, позволяющая пользователям загружать, просматривать и делиться видео.	*YouTube Data API v3*

Архитектура системы: Модули и взаимодействие

Модуль Bot

Отвечает за взаимодействие с пользователем через *Telegram*. Обрабатывает команды *(/track, /untrack, /list)*, валидирует ссылки, регистрирует пользователей и отправляет уведомления.

Модуль Scrapper

Выполняет основную бизнес-логику: периодически проверяет обновления на страницах, обрабатывает внешние *API (GitHub, StackOverflow, Habr, YouTube)* и *RSS*-ленты. Хранит данные в *PostgreSQL*.

Система построена по двухмодульной архитектуре, обеспечивающей слабую связанность компонентов. Взаимодействие между *Bot* и *Scrapper* осуществляется через *REST API*, что упрощает отладку и масштабирование. *Scrapper* имеет два подкомпонента: процедура обхода *(Scheduled Task)* и *REST API*.

Сценарии использования и технологии

Основные сценарии использования:

- **Регистрация пользователя:** Пользователь отправляет */start*, *Bot* регистрирует его в *Scrapper*.
- **Добавление ссылки:** Пользователь отправляет */track*, *Bot* валидирует и *Scrapper* сохраняет ссылку.
- **Удаление ссылки:** Пользователь отправляет */untrack*, *Scrapper* удаляет подписку.
- **Получение уведомлений:** *Scrapper* обнаруживает обновления и отправляет *POST*-запрос в *Bot*, который уведомляет пользователя.

Выбранные технологии:

Java & Spring Boot

Для высокопроизводительной и масштабируемой серверной части.

PostgreSQL

Для надежного хранения структурированных данных о пользователях и ссылках.

Flyway

Для эффективного управления миграциями базы данных.

Telegram API

Для удобного и широкодоступного пользовательского интерфейса бота.

Тестирование системы и заключение

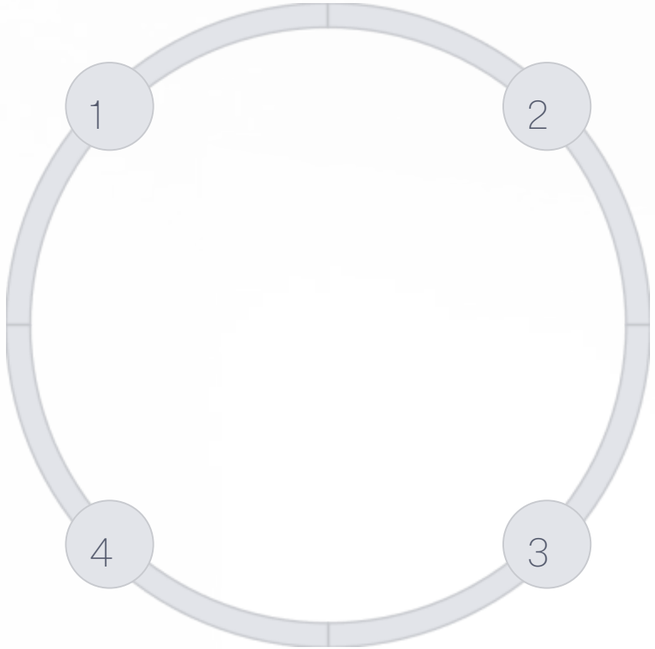
Тестирование подтвердило корректность работы обновленной системы и ее готовность к дальнейшему расширению.

Функциональное тестирование

Проверка добавления/удаления ссылок *(Habr, YouTube)*, получения списка, корректности уведомлений. Использовалось ручное тестирование бота и *REST*-запросы.

Автоматизация тестирования

Юнит-тесты *(JUnit 5, Mockito)* для логики сервисов *Scrapper* и командного процессора *Bot*. Интеграционные тесты с *Spring Boot Test* для *REST API*.



Интеграционные тесты

Проверка обмена данными между *Bot* и *Scrapper*, работы очереди обновлений и актуальности данных от внешних *API*. Использованы *Postman, WireMock*.

Тестирование интерфейсов

Фокус на понятности *UI Telegram*-бота, наличии инструкций и устойчивости к некорректным вводам. Ручное взаимодействие.

В ходе НИР была доработана система уведомлений, расширен перечень поддерживаемых платформ *(Habr, YouTube)*, улучшена стабильность и качество обработки запросов. Двухмодульная архитектура обеспечивает гибкость и масштабируемость, а проведенное тестирование подтвердило готовность системы к дальнейшему развитию.