Умные ссылки

Архитектура и шаблоны проектирования

Меня хорошо видно & слышно?

Защита проекта
Тема: Умные ссылки

Дмитрий Иванов

Ведущий разработчик и техлид распределенной команды в компании КубТри.
Email: hidiv@mail.ru

План защиты

- Цель и задачи проекта
- Какие технологии использовались
- Что получилось
- Выводы
- Вопросы и рекомендации

Цель и задачи проекта

Цель проекта: разработать гибкую систему «Умных ссылок», которая на основании настраиваемых правил автоматически перенаправляет пользователя на соответствующий целевой URL.

1. Спроектировать структуру хранения правил и действий в базе данных.
2. Реализовать механизм динамического разрешения и выполнения условий с поддержкой разных типов проверок.
3. Организовать механизм регистрации и вызова обработчиков действий через Dependency Injection и теги.
4. Обеспечить интеграцию всего конвейера: от приёма HTTP-запроса до выбора стратегии, проверки условий и выполнения редиректа с полным покрытием тестами.

Какие технологии использовались

1. **Symfony Framework** — обеспечивает структурированный подход к разработке, встроенный контейнер Dependency Injection и удобные инструменты для работы с HTTP-запросами и маршрутизацией.
2. **Doctrine ORM** — предоставляет надежную и гибкую абстракцию для хранения стратегий, условий и действий в базе.
3. **Dependency Injection + Tagged Services** — через теги легко регистрировать новые обработчики условий и действий без правки «ядра» системы.
4. **Codeception + PHPUnit** — для функционального и unit-тестирования всего конвейера, обеспечивая расчет коэффициента покрытия тестами и удобство тестирования.

Обзор решения

1. Цепочка middleware (Chain of Responsibility) для первичной фильтрации запросов.
2. Ядро «умных ссылок»: стратегии → условия → действия.
3. Хранение правил в БД: Strategy, Condition, Action.
4. Внедрение через Dependency Injection и теги.
5. Микросервис в Docker, готовый к CI/CD.

Архитектура и ключевые паттерны

1. **Strategy** — каждое условие и действие как отдельная стратегия.
2. **Chain of Responsibility** — последовательная обработка запросов.
3. **Repository** — получение стратегий, условий и действий из БД.
4. **Factory/Resolver** — динамическая подстановка по handlerTag.
5. Все компоненты соответствуют SOLID.

Расширяемость и поддержка

1. Новые условия/действия — просто добавить класс + тег.
2. Никаких правок «ядра» и изменения уже существующего кода.
3. Приоритеты тегов управляют порядком без изменения конфигурации.
4. DI-контейнер собирает все реализации автоматически.
5. Гибкая настройка через изменение записей в БД.

Тестирование и CI/CD

1. Юнит- и функциональные тесты на Codeception + PHPUnit.
2. Покрытие репозиториев, стратегий, middleware и REST-эндпоинтов.
3. Автозапуск в GitHub Actions, кэш Composer, отчёты на Codecov.
4. Возможность локального и удалённого запуска в Docker.
5. Быстрая обратная связь и защита от регрессий.

Что получилось

1. Исходный код проекта расположен на GitHub по адресу https://github.com/HiDiv/OtusSmartLinks.
2. Реализовано достаточное количество unit и функциональных тестов покрывающих все части разработки.
3. Проект упакован и работает как независимый микросервис в Docker-окружении, что обеспечивает легкий деплой и изоляцию от остальной инфраструктуры.
4. Применены паттерны **Strategy** (для условий и действий), **Chain of Responsibility** (для последовательного перебора request-хендлеров) и **Repository** (для доступа к данным).

Проблемы сложности и пути их решения

1. **Линейный обход middleware → замедление**
   Префильтрация по условиям и кеширование.
2. **Рост числа стратегий и приоритетов**
   Веб-UI с drag-and-drop приоритетов, фильтрами и симулятором прохождения.
3. **Отсутствие устойчивости к исключениям**
   Обёртки-спасатели (try/catch) + Monolog/Sentry + fallback-стратегии.
4. **Горизонтальное масштабирование под нагрузкой**
   Docker-микросервис + Redis-кеш стратегий + авто-масштабирование контейнеров.

Дальнейшее развитие

1. Добавить новые типы условий: гео-локация, внешняя аутентификация, пользовательские атрибуты.
2. Внедрить централизованное логирование и расширить обработку исключений.
3. Реализовать UI-панель для администрирования стратегий и условий.
4. Интеграция с другими микросервисами через REST и EventBus.

Вопросы и рекомендации

\+ если есть вопросы

— если вопросов нет

Спасибо за внимание!