УДК 004.05

Студ. А.А. Иванова
Науч. рук. ст. преп. Е.А. Блинова

# ОСНОВНЫЕ ПРИНЦИПЫ РЕФАКТОРИНГА ЛЕГАСИ КОДА В ПРОЕКТЕ DOMAIN.BY

Рефакторинг кода является одним из самых важных этапов разработки программного обеспечения, который позволяет улучшить качество кода и ускорить процесс разработки, а также сделать исполнение кода компилятором более быстрым.

В процессе работы над дипломной задачей для проекта domain.by, белорусским регистратором доменных имен, у меня появилась необходимость в работе с легаси кодом, который был написан более 15 лет назад. На сегодняшний день, в IT индустрии проблема рефакторинга легаси кода является как никогда актуальной, так как с развитием языков программирования, проектов, содержащий легаси код, становится все больше.

Легаси код – это программный код, который был написан давно и уже не соответствует современным стандартам разработки. Работа с легаси кодом может быть сложной, но важной задачей для многих разработчиков. Чтобы иметь возможность работать над поставленной задачей, мне понадобилось изучить тему рефакторинга легаси кода, после чего я успешно применила свои знания на практике.

Рассмотрим все принципы рефакторинга легаси кода, которые я вывела в ходе работы с ним.

1. Понимание кода. Перед тем, как начать работу с легаси кодом, необходимо понять, как он работает и какие функции он выполняет. Это является сложной задачей, так как код, который необходимо разобрать, написан давно и другими разработчиками. Этот этап является фундаментальным в процессе рефакторинга легаси кода. Необходимо разобраться в коде, проанализировать его структуру, а также выяснить, какие функции он выполняет.

В моем случае, код представлял собой несколько монолитных методов на языке c# размером около 5000 строк, методы, которые использовались внутри этих модулей могли ссылаться на другие, и чтобы понять их суть, необходимо было спускаться в глубину кода на 5-8 уровней. Весь код, над которым необходимо было провести функцию рефакторинга, выполнял задачи по подбору, регистрации и оплате не только доменов, но и услуг, предоставляемых компаний, в целом.

2. Тестирование. Перед тем, как менять легаси код, нужно убедиться, что он работает корректно, и выявить ситуации, при которых

он работает не корректно. Для этого необходимо создать тесты, которые проверят работу и нового, после рефакторинга, так и старого кода. Важно понимать, что тестирование, может быть, как автоматизированным или мануальным, так и с помощью плагинов или программ, в моем случае использовался Postman, мануальное тестирование и функция debug в среде разработки Visualstudio.

В ходе тестирования легаси кода я выявила баги, связанные с поиском доменных имен, ранее информация о свободных доменах третьего уровня собиралась только из одного хранилища, а надо было из трех, также при конвертации имен на кириллице в формат Punycode, конвертирование было корректным только в случае с именами, полностью состоящими из латинских символов (например, домен.бел), однако конвертер работал не корректно в случаях конвертации доменов с названием латиницей, а зоной на английском языке (например домен.by), эти баги являлись незначительными, однако при этом они влияли на имидж компании.

3. Разделение на модули. Легаси код, как правило, является монолитным запутанным кодом, чаще всего в коде, подобного рода есть повторения. Разделение его на модули, представленные методами, может помочь упорядочить код и легче понять его структуру. Необходимо разбить код на небольшие методы, где каждый метод отвечает за свой функционал.

В моем случае такое разделение привело к лучшему понимаю кода, а также к избавлению от излишнего кода, который не выполнялся в процессе работы методов.

4. Удаление дублирующегося кода. Дублирующий код часто приводит к ошибкам и усложняет понимание кода, а также к замедленной работе проекта. Поэтому одним из основных принципов рефакторинга легаси кода является удаление дублирующего кода и вынос его в отдельные методы, которые позже будут использоваться в разных частях проекта.

В процессе применение данного принципа, мне удалось сократить количество кода в 3 раза, что безусловно приводит к более легкому чтению кода в будущем.

5. Использование современных инструментов. В ходе рефакторинга легаси кода, я прибегала к плагину DuplicateDetector, который помог мне в поиске дублирующего кода и избавления от него и Postman для тестирования HTTP запросов для серверной части, также мною была использована программа для анализа LINQ запросов к базе данных – LinqPad, данная программа помогает оптимизировать LINQ запросы в базу данных с помощью возможности конвертера кода на

различных языках в SQL запрос и просмотра ответа на него. В основном помогает использование инструментов для анализа кода, поиска дубликатов и автоматического тестирования.

6. Постепенный рефакторинг. Изменение большого количества кода за один раз может привести к большим поломкам в разных, казалось бы, не связных, частях проекта, так как легаси код может быть связан с большим количеством других функций в проекте.

В моем случае код сразу отвечал за заказ и оплату сразу нескольких услуг, которые предоставляет платформа. Поэтому лучше начать с небольших изменений и постепенно улучшать код шаг за шагом. При постепенном рефакторинге кода не стоит забывать о постоянном тестировании не только нового кода, но и всего проекта в целом, чтобы избежать поломок разных, казалось бы, не связных, частей проекта.

Работа с легаси кодом может быть сложной, но важной задачей для многих разработчиков. Следуя этим принципам, процесс рефакторинга легаси кода будет проходить легче. С помощью рефакторинга кода, его можно сделать более читаемым, понятным, легко поддерживаемым и быстрее выполняемым.

В ходе рефакторинга кода в проекте domain.by, были устранены мелкие баги, связанные с поиском свободных доменов и конвертации доменных имен в формат Punycode, также получилось избавиться от большого количества повторяющегося кода и создать новый, более эффективный код.

Стоит отметить, что в ходе рефакторинга кода, мною были оптимизированы запросы в базу данных проекта, что привело к ускорению обработки запроса (ранее обработка запроса программой занимала 2 секунды в среднем, сейчас 0.3 секунды), также были оптимизированы запросы на сервера реестра доменных зон. Рефакторинг кода привел к оптимизации затрат ресурсов на исполнение кода на сервере, также избавление от легаси кода привело к его более легкому чтению и созданию более понятной документации к проекту.

## ЛИТЕРАТУРА

1. Физерс, М. Эффективная работа с унаследованным кодом / М. Физерс Москва: Вильямс, 2017. 23 – 169 с.

2. Мартин, Р. Чистый код. Создание, анализ и рефакторинг / Р. Мартин Санкт-Петербург: Питер, 2021. 150 – 287 с.