УДК 004.451:004.75

ДЕКОМПОЗИЦИЯ СЕРВИСА КРЕДИТНОГО СКОРИНГА С ПРИМЕНЕНИЕМ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Зарубин В.А. (ТПУ), **Дмитрийчук Д.И.** (ТПУ)
Научный руководитель – кандидат технических наук, доцент Демин А.Ю.
(ТПУ)

Введение. Системы кредитного скоринга играют ключевую роль в автоматизации выдачи займов, обеспечивая оперативную оценку платёжеспособности заёмщиков. Эти системы анализируют данные заявителей, учитывая множество факторов, включая кредитную историю, текущие обязательства, годовой доход и другие важные показатели[1]. Эффективность работы скорингового сервиса напрямую влияет на скорость обслуживания клиентов, уровень кредитных рисков и общую конкурентоспособность финансовой организации

Рассматриваемая система изначально была реализована в виде единого сервиса кредитного скоринга, что на первых этапах обеспечивало быстрое развертывание и разработку нового функционала. Со временем этот подход начал создавать серьезные трудности. Увеличение числа клиентов привело к росту нагрузки, замедлению обработки заявок и усложнению разработки.

Существуют разные стратегии решения этих проблем. В простых случаях монолит можно масштабировать горизонтально, но при росте нагрузки возникают ограничения из-за связности компонентов и узких мест в базе данных. Вертикальное масштабирование даёт временный прирост производительности, но быстро достигает предела[2]. Оптимизация кода и базы данных лишь отсрочивает проблемы, связанные с усложнением архитектуры. Модульный монолит, разбивающий систему на независимые модули с чёткими интерфейсами, упрощает поддержку, но остаётся монолитом, ограничивая масштабирование и усложняя обновления без перезапуска всей системы[3].

В рассматриваемой системе, для решения этой проблемы было принято решение о переходе к микросервисной архитектуре. Этот подход предполагает разбиение единого сервиса скоринга на несколько независимых компонентов, каждый из которых выполняет строго определённую функцию[4].

Основная часть. Для перехода от монолитной архитектуры к микросервисной было необходимо провести декомпозицию системы. Первым шагом стало определение ключевых функциональных блоков, которые можно было выделить без ущерба для общей логики работы скоринга. В процессе реорганизации системы были выделены сервисы компьютерного зрения, отвечающие за обработку изображений и документов, аналитические сервисы, сервисы чёрных списков и стоп-факторов, а также ключевой микросервис ядра, в котором развернута ML-модель скоринга, выполняющая расчёты кредитного рейтинга и принимающая финальное решение по каждой заявке.

Внедрение событийно-ориентированной архитектуры стало ключевым этапом перехода к микросервисам, так как оно позволило снизить связанность компонентов и обеспечить их независимую работу. В отличие от модели "запрос-ответ", где сервисы взаимодействуют напрямую, здесь используется публикация и обработка событий: один сервис (Producer) отправляет событие в брокер сообщений, а другие (Consumers) получают и обрабатывают его независимо. Такой подход позволяет нескольким сервисам реагировать на одно событие, повышая отказоустойчивость и масштабируемость системы[5]. В качестве брокера используется Apache Kafka, обеспечивающий асинхронное взаимодействие и снижение

задержек в обработке заявок.

Развёртывание микросервисов в Kubernetes обеспечило гибкость и производительность системы за счёт масштабирования реплик через балансировщик нагрузки, который позволяет равномерно распределять запросы и снижать нагрузку на компоненты. Микросервисная архитектура идеально сочетается с Kubernetes, так как каждый сервис может масштабироваться и обновляться отдельно, в отличие от монолита, где такие процессы требуют значительных ресурсов[6]. Инструменты мониторинга, такие как Prometheus, Grafana и Sentry, вместе с трейсингом Jaeger, позволяют в режиме реального времени отслеживать метрики, устранять ошибки и анализировать прохождение запросов, упрощая управление системой.

Исходный сервис скоринга продолжает выполнять роль прослойки между новой системой и основным корпоративным сервисом на Java. Новые микросервисы, созданные на основе современных технологий, таких как FastAPI, успешно взаимодействуют с исходным сервисом, написанным на aiohttp, что подтверждает гибкость микросервисного подхода и его способность работать с разнородными технологическими стеками.

Выводы. Переход на микросервисную архитектуру значительно повысил эффективность кредитного скоринга: скорость обработки заявок удвоилась, а независимость компонентов снизила нагрузку на основной сервис. Это также позволило выпускать еженедельные релизы и быстро устранять ошибки. Использование Kubernetes для распределения нагрузки и масштабирование реплик улучшили производительность. Микросервисы открывают возможности для применения различных технологий, например, Golang.

Однако переход потребовал крупных инвестиций в инфраструктуру и расширения команды. На начальных этапах монолитная архитектура была более оптимальной, а микросервисы стали оправданы с ростом нагрузки и усложнением логики. Выбор архитектуры зависит от потребностей системы и целей компании.

Список использованных источников:

1. Кредитный скоринг: что это и как работает // Alfabank: сайт. – URL: https://alfabank.ru/help/articles/credit-cards/kreditnyj-skoring/ (дата обращения: 08.02.2025)
2. Масштабирование нагрузки web-приложений // Хабр: сайт. – URL: https://habr.com/ru/articles/113992/ (дата обращения: 09.02.2025)
3. О сложности и монолитах // Хабр: сайт. – URL: https://habr.com/ru/companies/ruvds/articles/770262/ (дата обращения: 09.02.2025)
4. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга. – СПб: Питер, 2025. – 544 с.: ил – (Серия «Библиотека программиста»)
5. A. Rahmatulloh, F. Nugraha, R. Gunawan and I. Darmawan, «Event-Driven Architecture to Improve Performance and Scalability in Microservices-Based Systems,» 2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)m Bandung, Indonesia, 2022, pp. 01-06, doi: 10.1109/ICADEIS56544.2022.10037390.
6. Шумилов М.И. Оптимизация высоконагруженных веб-проектов с использованием микросервисной архитектуры // Universum: технические науки : электрон. научн. журн. 2024. 11(128). URL: https://7universum.com/ru/tech/archive/item/18560