Беляева К. В. Паттерн проектирования для разработки веб-приложений – BFF // Международный журнал информационных технологий и энергоэффективности. – 2022. – Т. 7 № 3(25) часть 3 с. 86–89

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.415.25

ПАТТЕРН ПРОЕКТИРОВАНИЯ ДЛЯ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ – BFF

Беляева К. В.
РТУ МИРЭА – Российский технологический университет, Москва, Россия(119454, Москва, пр. Вернадского, 78), e-mail: kaleriaa@bk.ru

Статья посвящена описанию взаимодействия программ между собой посредством API. С усложнением бэкенда, то есть использование микросервисного подхода, появляются трудности, так как фронтенду необходимо запоминать API каждого микросервиса. Для решения проблемы существует паттерн проектирования для разработки веб-приложений Backend for Frontend (BFF), который описан в данной статье, а также немного затронут паттерн API Gateway.Ключевые Социотехническая система, идентификация кластеров и анализ устойчивости, нечеткая когнитивная модель, нечеткое отношение взаимовлияния.

Ключевые слова: Паттерн, backed for frontend, веб-разработка, интерфейс, фронтенд, бэкенд, API, Gateway.

PATTERN FOR DEVELOPING BACKEND FOR FRONTEND (BFF) WEB APPLICATIONS

Belyaeva K. V.
RTU MIREA - Russian Technological University, Moscow, Russia (119454, Moscow, Vernadsky Ave., 78), e-mail: kaleriaa@bk.ru

The article is devoted to the description of the interaction of programs with each other through the API. With the complication of the backend, that is, the use of a microservice approach, difficulties arise, since the frontend needs to remember the API of each microservice. To solve the problem, there is a design pattern for developing Backend for Frontend (BFF) web applications, which is described in this article, and the API Gateway pattern is also slightly affected.

Keywords: pattern, backed for frontend, web development, interface, frontend, backend, API, Gateway.

В современном мире, где технологии развиваются стремительно, каждый третий человек использует информационные технологии в работе и повседневной жизни. С каждым днем появляются новые сервисы, разрабатываются новый функционал и обновляются приложения. По данным Росстата 85% жителей России используют интернет ежедневно. Служба веб-аналитики StatCounter опубликовала рейтинг популярности браузеров за 2022 год. Чаще остальных используют Google Chrome (66,64%), на втором месте Microsoft Edge (10,07%), на третьем – Safari от Apple (9,61%). Результат исследований подтверждает тот факт, что интернет является неотъемлемой частью жизни человека.

Огромное количество веб-сервисов и ресурсов покрывают разные потребности, но, к сожалению, удовлетворить абсолютно все нужды не представляется возможным с экономической и технической точки зрения. Так, благодаря API новые приложения

интегрируются с существующими программными системами, что, в конечном итоге, увеличивает скорость разработки, потому что каждую функцию не требуется писать самостоятельно с нуля. API (Application Programmable Interface) – это программный интерфейс, который позволяют двум программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов. В определении API интерфейс можно рассматривать как контракт между двумя программами, которая одна предоставляет другой.

Так, к примеру, система метеослужбы содержит ежедневные данные о погоде. Приложение погоды на телефоне или интегрированное на главную страницу браузера «обменивается» с этой системой через API и показывает ежедневные обновления погоды на телефоне. API позволяет настроить как разные компоненты программы должны эффективно взаимодействовать и используется повсеместно, что показывает важность данной технологии.

Современные веб-приложения реализуются с помощью графического интерфейса (фронтенда), с которым взаимодействует пользователь, и бэкенда, который скрыт от конечного пользователя и обеспечивает работу всего приложения, обрабатывая запросы от «клиента» и производя соответствующие действия.

В настоящее время частой практикой является разбиение приложения на отдельные небольшие приложения с ограниченной функциональностью – микросервисы, которые реализуют микросервисную архитектуру. Так, изначально данный подход использовали только на стороне бэкенда, но вскоре был реализован и на стороне фронтенда.

Среди многочисленных плюсов миркосервисной архитектуры есть и минусы, один из которых: каждый микросервис реализует уникальный программный интерфейс, поэтому фронтенд должен знать о каждом API, в связи с этим образуется жесткая сцепка между двумя сторонами. Также со стороны сервера могут поступать данные, не подходящие под нужды «клиента», допустим нефильтрованные данные, следовательно, браузер должен затратить больше ресурсов.

Решением данной проблемы является создание единого программного интерфейса, который будет знать о всех микросервисах, а также вынести несложную логику на отдельный уровень, которая обеспечивает контроль над данными. Этим промежуточным уровнем является BFF [3].

Backend for Frontend (BFF) – паттерн проектирования для разработки веб-приложений, основанный на идее API Gateway (единое окно), был разработан в компании SoundCloud. Gateway также является паттерном, так, оба этих шаблона используют один контракт для доступа к разным API. BFF используют в случаях, когда необходимо получать доступ от разных API и когда реализованы микросервисы.

Возможности BFF [4]:

- взаимодействовать с микросервисами и получать от них данные;
- преобразовать данные в соответствии с необходимым представлением;
- отправлять данные «клиенту».

Традиционный подход использует один gateway для всех «клиентов». Благодаря BFF покрываются потребности каждого «клиента», к примеру, мобильного, десктопного приложений, веб-сайта и т.д., так как возможно создать API для каждого. То есть возможность поддерживать несколько BFF позволяет избавиться от ограничений бэкенда и от хранения всего в одном месте. В связи с этим упрощается разработка сервисов, работающих с разными «клиентами».

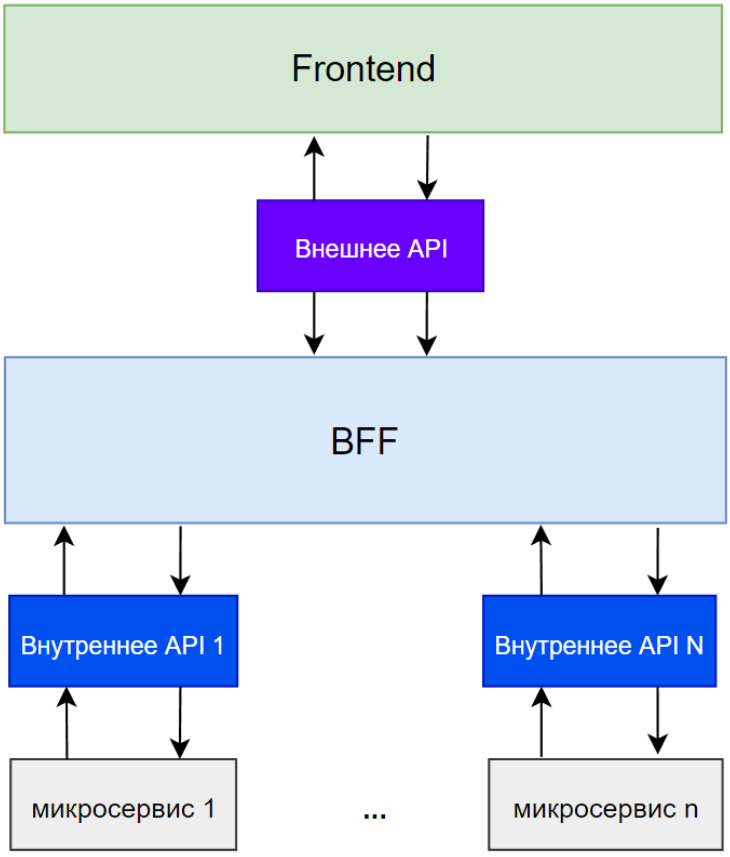

Рисунок 1 – Pattern Backend for Frontend

На рисунке 1 схематически изображен паттерн Backend for Frontend, который является промежуточным слоем между фронтендом и бэкендом, реализованным с помощью микросервисов.

Единое окно должно обеспечивать покрытие следующих требований [6]:

- высокую скорость ответа под нагрузкой;
- единый язык программирования, используемый на фронтенде и сервере BFF;
- единые шаблоны на «клиенте» и сервере.

Стоит отметить, что BFF должен быть реализован как простой интерфейс между фронтендом и бэкендом. Главным приоритетом является обмен данных.

Преимущества паттерна [5]:

- проще поддерживать и модифицировать API;
- бизнес-логика выносится на отдельный уровень;
- параллельно несколько «клиентов» могут обращаться к серверной части, что позволяет обслуживать сразу веб-сервисы и мобильные устройства, удерживая высокий уровень ответа;
- улучшенный пользовательский опыт – BFF обрабатывает ошибки сервера так, чтобы они были понятны пользователю;
- бэкенд разрабатывается на основе продукта;
- ускорение разработки.

Для того, чтобы паттерн приносил пользу, а не был во вред, необходимо понимать, в каких случаях приемлемо применять данную технологию. Ниже представлены основные критерии:

- на стороне бэкенда используется микросервисная архитектура;
- есть необходимость обслуживать несколько типов «клиентов» (мобильные, веб-приложения и т.д.);
- «клиенту» необходимо использовать данные, агрегирующиеся на стороне сервера.

Паттерн проектирование Backend for Frontend своеобразный переводчик между фронтендом и бэкендом, позволяющий облегчить и ускорить разработку, а также улучшить пользовательский опыт. Однако, не во всех случаях необходимо использовать данный стиль, так что перед решением внедрить, следует определить цели и понять, какие проблемы можно решить, применяя данный шаблон.

Список литературы

1. Что такое API? // AWS Amazon [Эл. ресурс]. URL: https://aws.amazon.com/ru/what-is/api/ (дата обращения: 25.11.2022).
2. Что такое API? // Habr URL: https://habr.com/ru/post/464261/ (дата обращения: 25.11.2022).
3. Паттерны Gateway и BFF // Дока URL: https://doka.guide/tools/gateway-bff/ (дата обращения: 26.11.2022).
4. Backend for Frontend BFF // Medium URL: https://medium.com/mobilepeople/backend-for-frontend-pattern-why-you-need-to-know-it-46f94ce420b0 (дата обращения: 26.11.2022).
5. The BFF Pattern (Backend for Frontend BFF) // Bits URL: https://blog.bitsrc.io/bff-pattern-backend-for-frontend-an-introduction-e4fa965128bf (дата обращения: 26.11.2022).
6. Backend for Frontend BFF: когда простого API не хватает // Habr URL: https://habr.com/ru/post/557406/ (дата обращения: 26.11.2022).

References

1. What is an API? // AWS Amazon [Email] resource]. URL: https://aws.amazon.com/en/what-is/api/ (Accessed 11/25/2022).
2. What is an API? // Habr URL: https://habr.com/ru/post/464261/ (date of access: 11/25/2022).
3. Gateway and BFF Patterns // Doka URL: https://doka.guide/tools/gateway-bff/ (accessed 11/26/2022).
4. Backend for Frontend BFF // Medium URL: https://medium.com/mobilepeople/backend-for-frontend-pattern-why-you-need-to-know-it-46f94ce420b0 (accessed 11/26/2022) .
5. The BFF Pattern (Backend for Frontend BFF) // Bits URL: https://blog.bitsrc.io/bff-pattern-backend-for-frontend-an-introduction-e4fa965128bf (accessed 11/26/2022).
6. Backend for Frontend BFF: when a simple API is not enough // Habr URL: https://habr.com/ru/post/557406/