Сравнительный анализ подходов backend-driven UI

Макаров Дмитрий Александрович
Старший разработчик мобильных приложений, Яндекс
Алматы, Казахстан
E-mail: dm.a.makarov@gmail.com

Аннотация: Статья посвящена изучению архитектурного подхода Backend-Driven UI (BDUI), предполагающего перенос управления пользовательским интерфейсом на серверную сторону. Актуальность темы определяется необходимостью повышения гибкости и сокращения сроков обновления приложений в условиях быстро меняющихся требований рынка. Научная новизна работы заключается в комплексном сравнении различных реализаций BDUI и выявлении их сильных и слабых сторон в сопоставлении с традиционными и гибридными методами построения интерфейсов. В рамках исследования описаны принципы формирования серверных UI-спецификаций, механизмы централизованного управления интерфейсами и специфика их внедрения в мультиплатформенных приложениях. Изучены преимущества, такие как ускорение итераций разработки, унификация пользовательского опыта и возможности A/B-тестирования, а также ограничения, связанные с усложнением серверной архитектуры и повышенными требованиями к сетевой надежности. Особое внимание уделено анализу практического опыта использования BDUI в индустрии и перспективам стандартизации описаний интерфейсов. Работа ставит цель систематизировать современные подходы к BDUI, используя сравнительный анализ и обзор актуальных источников. Применение результатов исследования будет полезно для разработчиков мобильных и веб-приложений, специалистов в области UX-инжиниринга и архитекторов распределенных систем.

Ключевые слова: Backend-Driven UI, server-driven UI, мобильные приложения, мультиплатформенные интерфейсы, динамическое обновление

Comparative analysis of backend-driven UI approaches

Dmitrii Makarov

Senior Mobile Application Developer, Yandex

Almaty, Kazakhstan

Email: dm.a.makarov@gmail.com

Abstract: The article is devoted to the study of the architectural approach of Backend-Driven UI (BDUI), which involves the transfer of user interface management to the server side. The relevance of the topic is determined by the need to increase flexibility and reduce the time required to update applications in the face of rapidly changing market requirements. The scientific novelty of the work lies in a comprehensive comparison of various BDUI implementations and the identification of their strengths and weaknesses in comparison with traditional and hybrid interface construction methods. The research describes the principles of the formation of server UI specifications, the mechanisms of centralized interface management and the specifics of their implementation in multiplatform applications. Advantages such as faster development iterations, unification of user experience and A/B testing capabilities, as well as limitations associated with the increasing complexity of the server architecture and increased requirements for network reliability are studied. Special attention is paid to the analysis of the practical experience of using BDUI in the industry and the prospects for standardization of interface descriptions. The work aims to systematize modern approaches to BDUI using comparative analysis and an overview

of relevant sources. The application of the research results will be useful for developers of mobile and web applications, specialists in the field of UX engineering and architects of distributed systems.

Keywords: Backend-Driven UI, server-driven UI, mobile applications, multiplatform interfaces, dynamic updating

Введение

В современных мобильных и мультиплатформенных приложениях все более актуальным становится подход Backend-Driven UI (BDUI), также известный как Server-Driven UI. Данный архитектурный подход предполагает перенесение управления пользовательским интерфейсом на сторону сервера. Актуальность BDUI обусловлена стремлением к повышению гибкости и скорости итераций разработки пользовательского опыта. Организации получают возможность быстрее вносить изменения в UI, проводить A/B-тестирование и обеспечивать единообразие интерфейсов на разных платформах.

Цель данной работы — проанализировать различные подходы к реализации Backend-Driven UI и сравнить их сильные и слабые стороны. В соответствии с поставленной целью в работе решаются следующие задачи:

— обзор концепции BDUI и ее разновидностей;

— сравнение архитектурных решений и инструментов, используемых для реализации BDUI;

— оценка преимуществ и ограничений каждого подхода;

— анализ практического опыта внедрения BDUI в индустрии.

Настоящее исследование носит актуальный характер, опираясь на новейшие данные и публикации (не старше пяти лет), и направлено на систематизацию знаний об эволюционном сдвиге в парадигме разработки UI.

Материалы и методы исследования

Для подготовки статьи использованы современные исследования, посвященные Backend-Driven UI и смежным архитектурным подходам. А.В. Гурин [1] исследовал специфику применения BDUI в iOS и показал его влияние на масштабируемость мобильных приложений. С.Д. Михайлюк [2] рассмотрел BDUI как стратегию оптимизации разработки и поддержки интерфейсов, выявив его потенциал для ускорения жизненного цикла продукта. M. Abdal, T. Mohamed, S. Jan и F.Q. Khan [3] провели сравнительный анализ различных методов разработки мобильных приложений, в том числе серверно-ориентированных решений. A. Carrera-Rivera, F. Larrinaga, G. Lasa и G. Martinez-Arellano [4] предложили фреймворк AdaptUI для построения адаптивных пользовательских интерфейсов в smart PSS, показав практическую ценность BDUI. V.N.S.K. Challa [5] выполнил обзор современных стратегий рендеринга приложений, уделив внимание преимуществам серверного управления интерфейсами. D. Pimenov, A. Solovyov, N. Askarbekuly и M. Mazzara [6] исследовали data-driven подходы к проектированию интерфейсов, акцентируя внимание на применимости динамической генерации UI. A.J. Undirwadkar [7] охарактеризовал BDUI как новый этап в мобильной разработке, подчеркнув его значимость для ускорения обновлений и экспериментов.

В процессе работы был проведён сравнительный анализ архитектурных решений, систематизация подходов и сопоставление их преимуществ и ограничений. Методологическая база включала анализ источников, интерпретацию инженерных практик и выделение типичных сценариев применения.

Результаты и обсуждения

Архитектурный паттерн Backend-Driven UI позволяет преодолеть ограничения традиционного

клиент-ориентированного подхода в разработке интерфейсов. В традиционных мобильных приложениях каждая модификация UI требует обновления клиентского кода и повторной публикации приложения. В отличие от этого, в BDUI сервер отвечает не только за данные, но и за описание того, какие UI-компоненты и в каком порядке должен отобразить клиент. По сути, сервер формирует «экран» из наборов компонентов, а клиентское приложение выступает в роли универсального рендерера этих компонентов. Подобное разбиение ответственности упрощает обновление интерфейса: разработчики могут вносить изменения на сервере и сразу видеть их отражение у пользователей, минуя цикл переустановки приложения.

Практическая реализация BDUI обычно предполагает, что на клиенте заложен некоторый базовый «контейнер» или движок рендеринга, способный отобразить широкий спектр компонентов интерфейса [4]. Все необходимые данные о структуре UI (экраны, разделы, элементы, их свойства и вложенность) передаются с сервера при запуске приложения или по запросу. Например, в рамках адаптивных пользовательских интерфейсов для умных сервисов (Smart PSS) предложен фреймворк, использующий паттерн server-driven UI: сервер высылает на клиент описания экранов (так называемые UID-дескрипторы), которые клиент динамически преобразует в визуальные компоненты интерфейса. Такой подход предоставляет большую гибкость, позволяя серверу генерировать и изменять UI-дескрипторы на лету в зависимости от контекста и предпочтений пользователя. Клиентское приложение при этом не требует переразвертывания — оно просто интерпретирует новые структуры и отображает их (Таблица 1).

Таблица 1. Классификация архитектурных решений Backend-Driven UI (составлено автором на основе [3–5])

Подход	**Принцип построения**	**Преимущества**	**Ограничения**	**Типичные сценарии применения**
Полный server-driven UI	Полная передача описания интерфейса с сервера	Унификация интерфейсов, централизованное управление	Высокие требования к серверу, сложность версионирования	Массовые приложения с частыми обновлениями
Частичный BDUI	На сервере описывается только часть структуры (контент, порядок блоков)	Гибкость в обновлениях контента, упрощение A/B-тестов	Ограниченные возможности изменения структуры	CMS, редакторские системы
Гибридный подход	Сервер задаёт layout, клиент отвечает за визуализацию	Баланс гибкости и производительности	Усложнение клиентской логики	Кроссплатформенные сервисы

Это обеспечивает режим реального времени для адаптации интерфейса, что особенно ценно для мультиплатформенных сервисов: изменения моментально отражаются во всех версиях (мобильной, веб и пр.) без выпуска обновлений.

Среди современных стратегий разработки мобильных приложений BDUI занял заметное место наряду с такими подходами, как нативный, гибридный и кроссплатформенный рендеринг.

В обзорной работе Server-Driven UI упоминается как одна из ключевых техник динамической генерации интерфейса на мобильных платформах, демонстрирующая свои преимущества в реальных кейсах крупных IT-компаний [5]. К числу основных достоинств BDUI относится возможность более быстрого и централизованного обновления контента. Команда разработки может оперативно вносить правки в серверную конфигурацию интерфейса и мгновенно прокатывать их на всех пользователях, без задержек на согласование и выпуск клиентских обновлений через магазины приложений. Это значительно ускоряет цикл итераций при экспериментах с UI и новыми фичами. Например, проведение A/B-тестов упрощается до минимальных изменений на бэкенде: разные группы пользователей могут получать разные варианты разметки интерфейса с сервера, тогда как само приложение остается единым и неизменным (Таблица 2).

Таблица 2. Сравнение Backend-Driven UI с альтернативными стратегиями разработки (составлено автором на основе [1, 4, 6])

Подход	**Скорость внедрения изменений**	**Зависимость от обновлений клиента**	**Уровень персонализации**	**Консистентность интерфейсов**
Нативный UI	Низкая (через магазины приложений)	Высокая	Средний	Ограниченная
Гибридный UI	Средняя	Средняя	Средний	Средняя
Кроссплатформенный UI	Средняя	Средняя	Средний	Выше среднего
Backend-Driven UI	Высокая (централизованно)	Низкая	Высокий	Высокая

Такой механизм повышает адаптивность продукта — можно быстро проверять гипотезы дизайна, персонализировать опыт под разные сегменты аудитории.

Еще одно важное преимущество — унификация пользовательского опыта на разных устройствах и платформах. Поскольку логика формирования интерфейса сконцентрирована на сервере, можно гарантировать консистентность отображаемых элементов на мобильных приложениях под iOS и Android, веб-клиентах и др.. Разработка новых фич при этом упрощается: вместо реализации отдельных интерфейсных решений под каждую платформу достаточно один раз задать их описание на сервере [1]. Например, если сервис доступен как в виде мобильного приложения, так и веб-сайта, подход BDUI позволяет синхронно обновлять UI и там, и там, давая пользователям единообразный функционал одновременно. Это также облегчает поддержку платформо-специфичных гайдлайнов — сервер может выдавать несколько вариаций разметки с учетом особенностей платформы (скажем, Material Design для Android и Human Interface для iOS), сохраняя при этом общий подход и логику.

Однако внедрение Backend-Driven UI связано и с определенными трудностями. Во-первых, возрастает сложность серверной части системы. Бэкенд фактически берет на себя роль, частично аналогичную фронтенду, что требует разработки специализированных модулей генерации UI-дескрипторов и их версионирования. Необходимость поддерживать обратную совместимость между

серверными описаниями интерфейса и различными версиями клиентских приложений — серьезный архитектурный вызов. Каждое изменение структуры UI на сервере должно аккуратно контролироваться, чтобы старые версии приложения не потеряли работоспособность при получении неизвестных им компонентов. Во-вторых, перераспределение обязанностей может привести к дисбалансу нагрузки между фронтендом и бэкендом. Поскольку логика отображения перемещается на сервер, нагрузка на серверную инфраструктуру растет — он должен обрабатывать запросы на UI и выдавать готовые конфигурации, тогда как клиент выполняет более тривиальную функцию рендеринга. Это требует высоконадежного и масштабируемого сервера, способного обслуживать потенциально большое число UI-запросов с минимальной задержкой, иначе теряется смысл мгновенных обновлений интерфейса.

Кроме того, ценой гибкости нередко становятся более высокие начальные затраты на разработку архитектуры BDUI [2]. Необходимо спроектировать универсальную модель представления интерфейса (например, формат JSON-схем), реализовать на клиенте абстрактный слой рендеринга и обеспечить безопасность и контроль ошибок при передаче описаний UI. По данным инженерной практики, изначальная сложность окупается в дальнейшем за счет ускорения развития продукта, однако на старте проектирования команде приходится инвестировать существенные ресурсы. Наконец, нельзя не упомянуть потенциальное снижение производительности или ухудшение UX при неосторожном использовании BDUI. Поскольку интерфейс формируется на основе ответов сервера, увеличивается зависимость от сети: при медленном соединении пользователь может дольше ждать отображения контента, особенно при первом запуске приложения, когда нужно загрузить все описание UI. Для смягчения этого эффекта применяют техники кеширования описаний на устройстве и заранее продуманные «скелетоны» интерфейса, отображаемые до прихода данных с сервера.

Таким образом, результаты анализа показывают, что Backend-Driven UI является мощным подходом для быстрого и гибкого управления интерфейсом приложений, обладая значительными преимуществами в части ускорения обновлений, экспериментирования и кроссплатформенной консистентности UI. Одновременно с этим, успешная реализация BDUI требует преодоления ряда инженерных вызовов, включая усложнение серверной логики, перераспределение нагрузки и обеспечение надежности взаимодействия клиент-сервер. Различные подходы к BDUI делают акцент на тех или иных аспектах (Таблица 3) [3,6].

Таблица 3. Подходы к реализации Backend-Driven UI и их особенности (составлено автором на основе [2, 6, 7])

Подход	**Основной акцент**	**Особенности реализации**	**Примеры использования**
Полностью серверный	Максимальная динамичность интерфейса	Генерация UI-дескрипторов на сервере	Крупные социальные сети, e-commerce
Постепенное внедрение	Контент и конфигурация вынесены на сервер	Использование feature toggle, remote config	Финтех-приложения, онлайн-сервисы
Гибридный	Баланс серверного и клиентского рендеринга	Сервер задаёт каркас, клиент – детали и анимации	Стриминговые платформы, мультимедиа-сервисы

Например, полностью серверный контроль UI реализован в таких решениях, как Airbnb Ghost или Yandex DivKit, где сервер формирует полное описание экранов, а клиент лишь выполняет рендеринг компонентов. Постепенное внедрение удалённой конфигурации характерно для крупных мобильных приложений — например, в финтех-сервисах (Revolut, Tinkoff), где изменение интерфейса осуществляется через механизмы remote config и feature toggle без необходимости обновления клиента. Гибридный подход используется в Netflix и ряде мультимедийных платформ: сервер задаёт layout и наполняет контентом, а окончательная визуализация, анимации и нативные элементы остаются за клиентом (iOS или Android). Подобные кейсы показывают, что BDUI применяется по-разному: от полного контроля на стороне сервера до частичного вынесения конфигураций или комбинированных схем. В следующих разделах обсуждения будут рассмотрены ситуации, в которых оправдан тот или иной вариант, и перспективы дальнейшей стандартизации описаний интерфейсов.

Полученные результаты подтверждают, что переход к Backend-Driven UI отражает более общую тенденцию возврата части логики с клиента на сервер, продиктованную требованиями гибкости и скорости обновлений. Исторически в индустрии наблюдается «маятниковый» процесс перераспределения ответственности между фронтендом и бэкендом. Появление Web 2.0 и распространение мобильных приложений изначально сместили баланс в сторону толстых клиентов с богатой локальной логикой. Однако рост потребности в оперативном управлении интерфейсом и персонализации опыта привели к обратному движению: архитектуры вроде BDUI возвращают часть контроля серверу, фактически предлагая современную интерпретацию идей тонкого клиента. Анализ показал, что BDUI особенно эффективен в сценариях, где требуются частые изменения UI, проведения экспериментов и наличие множества платформ. Для крупных сервисов с миллионами пользователей возможность мгновенно переключить конфигурацию интерфейса централизованно дает огромные выигрыши в скорости развёртывания изменений. Это приводит к сокращению цикла разработки нового функционала с недель до часов [7].

С другой стороны, не каждый проект нуждается в Backend-Driven UI. Если приложение относительно простое, с редкими обновлениями интерфейса и небольшим числом экранов (например, музыкальные редакторы вроде GarageBand, дневники или персональные трекеры), усложнение архитектуры может оказаться избыточным. Для принятия решения о внедрении BDUI полезно задать себе несколько вопросов:

— насколько часто требуется менять интерфейс и запускать A/B-тесты?

— есть ли критическая потребность в единообразии UI на разных платформах (iOS, Android, Web)?

— готова ли команда к созданию и поддержке собственного UI-сервера и протоколов описания интерфейсов?

— оправдывают ли потенциальные выгоды усложнение архитектуры и рост нагрузки на сервер?

Если хотя бы на часть этих вопросов ответ положителен, BDUI может дать значительные преимущества. Но если интерфейс редко меняется, а жизненный цикл приложения спокоен, классические подходы будут более рациональны. Нередко компании, решившиеся на BDUI, создают внутренние фреймворки и библиотеки — примером служит платформа Ghost в Airbnb, позволявшая формировать интерфейсы «на лету» для разных клиентов. Подобные кейсы показывают жизнеспособность концепции, но подчёркивают, что её внедрение — это серьёзный инженерный проект, оправданный не во всех сценариях.

Сравнивая подходы BDUI, можно отметить, что они лежат на спектре от полностью серверно-

ориентированных к гибридным. Полноценный Server-Driven UI в чистом виде предполагает, что практически весь UI описан на сервере, а клиент лишь отображает компоненты. Промежуточные подходы могут ограничиваться вынесением на сервер только части аспектов — например, текстового контента и порядка блоков, но без детализации мелких виджетов. Такой частичный BDUI иногда реализуется через системы управления контентом (CMS) или удаленные конфигурации, что упрощает редакторам и продуктовым менеджерам изменение содержимого экранов без участия разработчиков. Тем не менее, истинный BDUI идёт дальше, предоставляя возможность изменять саму структуру UI.

Внедрение BDUI способствует более тесному взаимодействию команд бэкенда и фронтенда. Грань между этими ролями размывается: для успешного результата серверные разработчики должны учитывать принципы UX-дизайна, а мобильные разработчики — предоставить достаточную универсальность клиентского рендерера. Возникает необходимость стандартизации «языка» описания интерфейсов — будь то собственный JSON-протокол, GraphQL-схемы или иные форматы. Неслучайно некоторые индустриальные решения для BDUI опираются на GraphQL, позволяя клиентам запрашивать не только данные, но и представление о том, как эти данные должны быть визуализированы. Такой подход, в частности, использует Netflix, сочетая серверный контроль UI с мощью GraphQL для формирования адаптивных лент контента.

В целом, обсуждая различные реализации BDUI, можно заключить, что этот подход гармонично дополняет существующие парадигмы разработки интерфейсов. Он не претендует на замену всех клиентских технологий, а предлагает инструмент для тех случаев, когда гибкость и скорость критически важны. Вполне вероятно, что в будущем мы увидим дальнейшее развитие идей Backend-Driven UI — например, появление стандартизированных фреймворков, поддерживаемых сообществом (уже сейчас открыты исходные коды подобных библиотек, например Yandex DivKit и др.). Научная и практическая значимость темы BDUI состоит в том, что она объединяет знания об архитектуре распределенных систем, UX-инжиниринге и мобильной разработке, предлагая новый уровень управляемости пользовательским опытом.

Заключение

Backend-Driven UI представляет собой инновационное направление в развитии архитектур интерфейсов, обеспечивающее высокую гибкость и скорость изменений за счет переноса управления на сервер. Сравнительный анализ показал его преимущества: централизованное обновление контента и дизайна без выпуска новых версий приложений, ускорение экспериментов и обеспечение единообразного пользовательского опыта на разных платформах. Концепция «UI как сервис» делает интерфейс столь же динамичным, как данные, что особенно ценно для компаний, которым требуется оперативная реакция на запросы рынка.

Вместе с тем выявлены ограничения: усложнение серверной части, рост требований к стабильности сети и необходимость тщательно проектировать структуру UI-дескрипторов. Подход оправдан прежде всего в крупных проектах с частыми изменениями интерфейса, где выгоды от ускорения вывода функций на рынок перекрывают издержки. Для небольших приложений более уместны традиционные методы.

Итоговый вывод состоит в том, что BDUI отражает объективный запрос на адаптивность современных приложений. Разнообразие его реализаций — от полного вынесения UI на сервер до гибридных схем — позволяет организациям выбирать подходящий уровень «серверной ориентированности». Дальнейшие исследования связаны с разработкой стандартов описания интерфейсов, механизмов согласованности клиент-серверных состояний и изучением влияния BDUI на качество пользовательского опыта.

Список литературы:

1. Гурин А. В. Backend-Driven UI для iOS: методология динамического обновления контента и ее влияние на масштабируемость мобильных приложений // Актуальные исследования. 2022. № 25 (104). URL: https://apni.ru/article/4310-backend-driven-ui-dlya-i-os-metodologiya-dinamicheskogo-obnovleniya-kontenta-i-ee-vliyanie-na-masshtabiruemost-mobilnyh-prilozhenij (дата обращения: 22.09.2025).

2. Михайлюк С. Д. SERVER-DRIVEN UI КАК СТРАТЕГИЯ ОПТИМИЗАЦИИ РАЗРАБОТКИ И ПОДДЕРЖКИ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ // Вестник науки. 2025. № 4 (85), С. 401–420. URL: https://cyberleninka.ru/article/n/server-driven-ui-kak-strategiya-optimizatsii-razrabotki-i-podderzhki-polzovatelskih-interfeysov (дата обращения: 23.09.2025).

3. Abdal M., Mohamed T., Jan S., Khan F. Q. A Comparative Analysis of Mobile Application Development Approaches // Proceedings of the Pakistan Academy of Sciences. 2021. Vol. 58, № 1. P. 35–45. DOI: 10.53560/PPASA(58-1)717.

4. Carrera-Rivera A., Larrinaga F., Lasa G., Martinez-Arellano G. AdaptUI: A Framework for the development of Adaptive User Interfaces in Smart Product-Service Systems // User Modeling and User-Adapted Interaction. 2024. Vol. 34, № 5. P. 1929–1980. DOI: 10.1007/s11257-024-09414-0. License: CC BY 4.0.

5. Challa V. N. S. K. Comprehensive Analysis of Modern Application Rendering Strategies: Enhancing Web and Mobile User Experiences // Journal of Engineering and Applied Sciences Technology. 2022. Vol. 4(4), P. 1-6. DOI: 10.47363/JEAST/2022(4)248.

6. Pimenov D., Solovyov A., Askarbekuly N., Mazzara M. Data-Driven Approaches to User Interface Design: A Case Study // Journal of Physics Conference Series. 2021. Vol. 2134, № 1. Article 012020. DOI: 10.1088/1742-6596/2134/1/012020. License: CC BY 3.0.

7. Undirwadkar A. J. The rise of server-driven UI: A paradigm shift in mobile app development // World Journal of Advanced Engineering Technology and Sciences. 2025. Vol. 15, № 2. P. 055–061. DOI: 10.30574/wjaets.2025.15.2.0538.