Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

УДК 004.4'2: 339.17 https://doi.org/10.33619/2414-2948/128/09

РАЗРАБОТКА АДАПТИВНОГО ВЕБ-ИНТЕРФЕЙСА НА ОСНОВЕ REACT.JS И TYPESCRIPT КАК ФАКТОР ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ЭЛЕКТРОННОЙ КОММЕРЦИИ КОМПЬЮТЕРНОЙ ТЕХНИКИ

©Аркабаев Н. К., *ORCID: 0009-0000-1912-2225, SPIN-код: 9304-5193,*
канд. физ.-мат. наук, Ошский государственный университет,
г. Ош, Кыргызстан, narkabaev@oshsu.kg
©Айтматова Г. Д., *ORCID: 0009-0008-6783-6431, Ошский государственный университет,*
г. Ош, Кыргызстан, aitmatova0101@gmail.com
©Аскарова Л. С., *ORCID: 0009-0005-2060-4519, Ошский государственный университет,*
г. Ош, Кыргызстан, askarovaliliya08@gmail.com

DEVELOPMENT OF AN ADAPTIVE WEB INTERFACE BASED ON REACT.JS AND TYPESCRIPT AS A FACTOR IN IMPROVING THE EFFICIENCY OF E-COMMERCE FOR COMPUTER EQUIPMENT

©Arkabaev N., *ORCID: 0009-0000-1912-2225, SPIN-code: 9304-5193,*
Ph.D., Osh State University, Osh, Kyrgyzstan, narkabaev@oshsu.kg
©Aitmatova G., *ORCID: 0009-0008-6783-6431, Osh State University,*
Osh, Kyrgyzstan, aitmatova0101@gmail.com
©Askarova L., *ORCID: 0009-0005-2060-4519, Osh State University,*
Osh, Kyrgyzstan, askarovaliliya08@gmail.com

Аннотация. Статья посвящена вопросам проектирования и разработки адаптивного веб-интерфейса системы электронной коммерции компьютерной техники с применением библиотеки React.js и языка TypeScript. Рассматривается взаимосвязь между качеством пользовательского интерфейса и коммерческими показателями интернет-магазина, прежде всего коэффициентом конверсии и средним временем оформления заказа. В работе обоснован выбор технологического стека для фронтенд-разработки, описана архитектура компонентов приложения и реализация адаптивной верстки с учетом специфики каталога компьютерной техники. Проведена сравнительная оценка производительности приложения до и после внедрения адаптивного интерфейса на основе React.js. Результаты показали, что переход на компонентную архитектуру с использованием React.js и TypeScript позволил существенно сократить время загрузки страниц каталога, повысить конверсию мобильных пользователей и снизить показатель отказов. Полученные данные подтверждают, что грамотный выбор инструментов фронтенд-разработки может оказывать непосредственное влияние на эффективность продаж в сегменте компьютерной техники.

Abstract. The article deals with the design and development of an adaptive web interface for a computer equipment e-commerce system using the React.js library and the TypeScript language. The relationship between user interface quality and the commercial performance of an online store is examined, with a particular focus on conversion rate and average order completion time. The study justifies the choice of the frontend technology stack, describes the component architecture of the application, and the implementation of an adaptive layout tailored to the specifics of a computer equipment catalog. A comparative evaluation of application performance was carried out before and after the introduction of the React.js-based adaptive interface. The results showed that the transition

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

to a component-based architecture using React.js and TypeScript led to a significant reduction in catalog page loading times, an increase in mobile user conversion, and a decrease in the bounce rate. The findings confirm that a well-informed choice of frontend development tools can have a direct impact on sales efficiency in the computer equipment segment.

Ключевые слова. адаптивный веб-интерфейс, электронная коммерция, компьютерная техника, компонентная архитектура.

Keywords. adaptive web interface, e-commerce, computer equipment.

Рынок электронной коммерции в странах Центральной Азии переживает период интенсивного роста. По данным отраслевых аналитиков, за период с 2021 г по 2024 г объем онлайн-торговли в Кыргызстане увеличился почти вдвое, причем сегмент компьютерной техники и комплектующих демонстрирует одну из наиболее высоких динамик [1].

Вместе с тем большинство региональных интернет-магазинов продолжают использовать устаревшие подходы к построению пользовательских интерфейсов, что негативно сказывается на коммерческих показателях и, в конечном счете, на конкурентоспособности торговых площадок. Особенность торговли компьютерной техникой состоит в том, что покупатель, как правило, оперирует большим количеством технических характеристик, сравнивает десятки позиций и ожидает от интерфейса быстрой фильтрации, удобного отображения спецификаций и бесперебойной работы на любом устройстве [2].

Если интерфейс не справляется с этими задачами, пользователь уходит к конкуренту, и никакая ценовая политика этого не компенсирует. По оценкам Baymard Institute, средний показатель отказов от корзины в мировом e-commerce превышает 69%, и значительная часть отказов связана именно с проблемами юзабилити (https://707.su/Cbxo).

Современные фреймворки фронтенд-разработки предлагают инструментарий, способный кардинально изменить ситуацию. Библиотека React.js, разработанная Meta (ранее Facebook) и широко применяемая в мировой практике создания веб-приложений, опирается на компонентную модель, виртуальный DOM и декларативный подход к описанию пользовательских интерфейсов [3].

В сочетании с языком TypeScript, который обеспечивает статическую типизацию и повышает надежность кодовой базы, React.js формирует стек, отвечающий требованиям промышленной фронтенд-разработки [4].

Вместе с тем в русскоязычной научной литературе вопрос о влиянии выбора конкретного фреймворка на коммерческую эффективность интернет-магазина исследован фрагментарно. Большинство публикаций ограничиваются техническим описанием фреймворков без привязки к бизнес-метрикам, либо рассматривают e-commerce в целом, не учитывая специфику товарных категорий с высокой информационной нагрузкой [5].

Это создает пробел, который данная статья призвана частично восполнить. Цель работы заключается в проектировании и реализации адаптивного веб-интерфейса интернет-магазина компьютерной техники на базе React.js и TypeScript с последующей оценкой его влияния на ключевые коммерческие показатели.

Материал и методы исследования

Настоящее исследование проводилось на базе действующего интернет-магазина компьютерной техники, расположенного в городе Ош (Кыргызстан). Магазин функционирует с 2015 г и обслуживает как розничных, так и мелкооптовых покупателей. Каталог на момент

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

исследования содержал более 1200 товарных позиций, разбитых по 14 основным категориям: процессоры, видеокарты, материнские платы, оперативная память, накопители (SSD и HDD), блоки питания, корпуса, системы охлаждения, мониторы, периферия, сетевое оборудование, кабели и адаптеры, программное обеспечение и готовые сборки.

Исходная версия интернет-магазина была построена на CMS WordPress с плагином WooCommerce и темой, разработанной сторонним подрядчиком в 2020 году. Интерфейс не был адаптирован полностью под мобильные устройства: на экранах шириной менее 768 пикселей элементы фильтрации, таблицы характеристик и карточки товара — отображались с горизонтальной прокруткой, кнопки добавления в корзину перекрывались другими элементами, а оформление заказа требовало от мобильного пользователя в среднем 11 нажатий (против 6 на десктопе). По данным внутренней аналитики (Яндекс.Метрика) доля мобильного трафика составляла 58%, но конверсия мобильных пользователей в 3,4 раза ниже десктопной.

Для решения обозначенных проблем была спроектирована новая клиентская часть приложения на React.js (18.2) и TypeScript (5.3). Серверная часть (API на Node.js с базой данных PostgreSQL) – осталась без существенных изменений. Это позволило удобно изолировать эффект от обновления фронтенда.

Методологическая рамка работы включала несколько компонентов. На этапе проектирования применялся метод декомпозиции интерфейса на независимые React-компоненты – архитектурный подход, сделанный с опорой на принцип единственной ответственности (Single Responsibility Principle). В качестве хранилища состояния приложения использовалась комбинация локального состояния (React useState/useReducer) для компонентного уровня и глобального стора на базе Zustand для сквозных данных (корзина, авторизация, фильтры каталога) [6]. Связывание между страницами было организовано через React Router v6.

Адаптивность интерфейса достигалась за счет введения трехуровневой системы breakpoints: для мобильных устройств (до 576 px), планшетов (577-992 px) и десктопа (от 993 px). Стилизация осуществлялась с помощью CSS-модулей, что исключает конфликты имен и изолирует стили отдельно на уровне компонентов. Для сложных таблиц спецификаций была реализована механика горизонтального свайпа с визуальной индикацией прокрутки, что существенно отличается от простого overflow-x: scroll, который игнорирует контекст взаимодействия на сенсорных устройствах.

Оценка эффективности нового интерфейса проводилась методом A/B-тестирования в течение 8 недель (с 15 января по 10 марта 2025 года). Трафик равномерно делился между старой (контрольная группа, n=4280 сеансов) и новой (экспериментальная группа, n=4350 сеансов) версиями сайта. Среди отслеживаемых метрик: конверсия (отношение числа оформленных заказов к числу уникальных сеансов), показатель отказов (bounce rate), среднее время загрузки страницы (по данным Lighthouse), среднее число просмотренных страниц за сеанс и средний чек. Статистическая значимость различий проверялась критерием хи-квадрат для конверсии и U-критерием Манна-Уитни для непрерывных метрик ($p<0.05$).

Результаты и их обсуждение

Архитектура приложения. Спроектированное React-приложение включает 47 переиспользуемых компонентов, сгруппированных в четыре уровня: UI-примитивы (Button, Input, Badge, Spinner – 12 компонентов), составные компоненты (ProductCard, FilterPanel, SpecTable, CartItem – 15 компонентов), страничные компоненты (CatalogPage, ProductPage, CartPage, CheckoutPage и др. – 9 компонентов) и контейнеры-обертки (Layout, AuthGuard, ErrorBoundary – 11 компонентов). Такая иерархия обеспечивает масштабируемость:

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

добавление новой товарной категории не требует написания нового кода для отображения, достаточно описать конфигурацию фильтров в JSON-формате.

Типизация через TypeScript позволила формализовать контракты между компонентами и существенно сократить число ошибок на этапе разработки. Для каждой сущности предметной области (Product, Category, CartItem, Order) были определены интерфейсы, что не только помогло при написании кода, но и послужило своеобразной документацией для членов команды. Ниже приведен фрагмент типизации основной сущности каталога:

```
// types/product.ts
export interface Product {
  id: string;
  name: string;
  category: CategorySlug;
  price: number;
  oldPrice?: number;
  images: string[];
  specs: Record<string, string | number>;
  inStock: boolean;
  rating: number;
  reviewCount: number;
}
export type CategorySlug =
  | "cpu" | "gpu" | "motherboard" | "ram"
  | "ssd" | "hdd" | "psu" | "case"
  | "cooling" | "monitor" | "periphery"
  | "network" | "cables" | "builds";
```

Каждое поле интерфейса Product строго типизировано, а поле specs использует обобщенную запись Record<string, string | number>, что позволяет единообразно обрабатывать спецификации товаров разных категорий – от тактовой частоты процессора до длины кабеля. Опциональное поле oldPrice (с модификатором?) появляется только у товаров со скидкой, и компонент ProductCard учитывает это при отрисовке, отображая зачеркнутую цену рядом с актуальной.

Реализация адаптивного интерфейса. Ключевым вызовом при проектировании адаптивной версии стала работа с каталогом, который содержит большое количество фильтров и табличных данных. На десктопе панель фильтров расположена слева от списка товаров и занимает примерно 25% ширины экрана. На мобильных устройствах эта панель трансформируется в выдвижной drawer, вызываемый по нажатию кнопки «Фильтры» в верхней части экрана. Переход между режимами реализован через хук useMediaQuery, который отслеживает текущую ширину viewport и передает соответствующее значение через контекст React.

Для карточки товара в каталоге разработаны два варианта отображения. В десктопном режиме карточка содержит миниатюру изображения, название, краткие характеристики (процессор: ядро, частота; видеокарта: объем памяти, тип; и т.д.), цену, рейтинг и кнопки «В корзину» и «Сравнить». В мобильном режиме карточка упрощается: характеристики убираются, остаются только изображение, название, цена и кнопка добавления в корзину, а доступ к подробным характеристикам предоставляется при переходе на страницу товара. Это решение было принято после анализа поведения мобильных пользователей, который показал,

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

что менее 8% из них используют краткие характеристики в каталоге для принятия решения о покупке.

Особого внимания заслуживает реализация компонента сравнения товаров. На десктопе таблица сравнения отображается в классическом табличном виде с фиксированным первым столбцом (названия параметров). На мобильных устройствах та же информация представлена в формате каскадных карточек: пользователь видит один товар целиком и может переключаться между товарами свайпом, при этом различия в характеристиках визуально выделяются цветом. Такой подход принципиально отличается от простого масштабирования десктопной таблицы и учитывает когнитивные особенности восприятия информации на малых экранах [7].

Таблица 1

СРАВНЕНИЕ КЛЮЧЕВЫХ МЕТРИК ИНТЕРФЕЙСА ДО И ПОСЛЕ ВНЕДРЕНИЯ React.js
(A/B-тестирование, 8 недель)

Метрика	*WooCommerce (контроль)*	*React.js + TS (эксперимент)*	*Изменение, %*
Конверсия общая	2,1%	2,8%	+33,3
Конверсия мобильных	0,9%	1,1%	+22,2
Bounce rate	54,3%	44,6%	–18,0
Время загрузки (мобильные), с	4,8	3,0	–37,5
Страниц за сеанс	3,2	4,7	+46,9
Средний чек, сом	4 520	4 890	+8,2

Анализ коммерческих метрик. Данные, полученные в ходе A/B-тестирования, свидетельствуют о статистически значимом улучшении по всем отслеживаемым показателям (Таблица 1). Общая конверсия выросла с 2,1% до 2,8%, что в абсолютных цифрах дало примерно 30 дополнительных заказов за период тестирования. Наиболее заметное улучшение зафиксировано в подгруппе мобильных пользователей: конверсия увеличилась с 0,9% до 1,1% (p=0,023). Хотя в процентных пунктах прирост кажется скромным, в относительном выражении он составляет 22,2%, и для бизнеса с ежемесячным мобильным трафиком в 2500-3000 сеансов это эквивалентно 5-6 дополнительным заказам в месяц. Показатель отказов снизился с 54,3% до 44,6%. Основной вклад в это снижение внесла улучшенная скорость загрузки: медианное время полной загрузки страницы каталога на мобильных устройствах сократилось с 4,8 до 3,0 секунды. По данным Lighthouse, индекс Performance вырос с 42 до 78 баллов, прежде всего за счет code splitting (React.lazy + Suspense), оптимизации изображений через формат WebP и ленивой загрузки компонентов, находящихся за пределами видимой области экрана. Количество просмотренных страниц за сеанс увеличилось с 3,2 до 4,7. Это можно объяснить двумя причинами: во-первых, переходы между страницами в SPA-приложении происходят без полной перезагрузки (клиентская маршрутизация), во-вторых, улучшенная система фильтрации и рекомендаций стимулирует пользователей просматривать больше товаров. Средний чек вырос на 8,2%, предположительно, отчасти благодаря новому блоку «Сопутствующие товары», который реализован как React-компонент с динамической подгрузкой рекомендаций на основе текущей категории.

Технические аспекты производительности. Помимо бизнес-метрик, были оценены технические показатели производительности приложения. Для измерений использовались инструменты Chrome DevTools и Google Lighthouse (версия 11.4). Результаты сведены в Таблицу 2.

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

Таблица 2

ПОКАЗАТЕЛИ ПРОИЗВОДИТЕЛЬНОСТИ ПО Google Lighthouse

Показатель	*WooCommerce*	*React.js + TS*
Performance Score	42	78
First Contentful Paint, с	3,2	1,4
Largest Contentful Paint, с	5,6	2,8
Time to Interactive, с	7,1	3,3
Cumulative Layout Shift	0,28	0,04
Размер бандла (gzip), КБ	1 240	380

Одной из причин значительного улучшения показателя Cumulative Layout Shift (с 0,28 до 0,04) стало использование фиксированных размеров для контейнеров изображений товаров. В исходной версии изображения загружались без указания атрибутов width и height, что приводило к «прыжкам» контента при их загрузке. В React-версии для каждого компонента ProductCard задаются резервные размеры через CSS-свойство aspect-ratio, а для lazy-loaded изображений используется плейсхолдер в виде размытой миниатюры (техника LQIP – Low-Quality Image Placeholder).

Размер клиентского бандла удалось сократить более чем в три раза – с 1240 до 380 КБ (gzip). Это стало возможным благодаря нескольким факторам: code splitting по маршрутам (каждая страница загружается отдельным чанком), tree shaking неиспользуемого кода через Webpack 5, отказ от jQuery и его плагинов (которые в WooCommerce-версии занимали около 340 КБ) и переход на нативные API браузеров (IntersectionObserver вместо плагинов скролл-анимаций, Fetch API вместо axios для HTTP-запросов).

Фрагмент реализации ленивой загрузки страниц каталога:

```
import { lazy, Suspense } from 'react';
import { Routes, Route } from 'react-router-dom';
import { Spinner } from './components/ui/Spinner';
const CatalogPage = lazy(() => import('./pages/CatalogPage'));
const ProductPage = lazy(() => import('./pages/ProductPage'));
const CartPage = lazy(() => import('./pages/CartPage'));
const CheckoutPage = lazy(() => import('./pages/CheckoutPage'));
export const App = () => (
  <Suspense fallback={<Spinner fullPage />}>
    <Routes>
      <Route path="/catalog/:category?" element={<CatalogPage />} />
      <Route path="/product/:id" element={<ProductPage />} />
      <Route path="/cart" element={<CartPage />} />
      <Route path="/checkout" element={<CheckoutPage />} />
    </Routes>
  </Suspense>
);
```

Благодаря React.lazy и Suspense при первоначальной загрузке пользователь получает только код, необходимый для текущей страницы. Страницы каталога и карточки товара, которые являются наиболее посещаемыми (82% трафика), загружаются приоритетно, а менее востребованные разделы (корзина, оформление заказа) подгружаются по мере необходимости.

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

Компонент Spinner служит индикатором загрузки, что улучшает воспринимаемую производительность и позволяет пользователю понять, что контент загружается, а не что приложение зависло.

Обсуждение результатов

Полученные результаты согласуются с выводами ряда исследователей, отмечавших положительное влияние адаптивного дизайна и компонентных фреймворков на коммерческие показатели веб-приложений [8, 9].

В частности, прирост конверсии в 22–33% сопоставим с данными, представленными в работах по оптимизации интернет-магазинов в сегменте электроники, хотя прямое сравнение затруднено различиями в исходных условиях и целевых аудиториях.

Заслуживает внимания вопрос о том, какую именно долю наблюдаемого эффекта можно отнести к технологии фронтенда как таковой, а какую – к сопутствующим UX-решениям (улучшенная фильтрация, блок рекомендаций, редизайн карточки товара). Строго разделить эти факторы в рамках проведенного эксперимента затруднительно, поскольку новый интерфейс разрабатывался как целостная система. Тем не менее, показатели, непосредственно связанные с технологией (время загрузки, размер бандла, CLS), демонстрируют однозначное улучшение, не зависящее от дизайнерских решений.

Нужно также отметить ограничения проведенного исследования. Во-первых, A/B-тест проводился в зимний период, который для рынка компьютерной техники характеризуется повышенным спросом (подготовка к учебному семестру, обновление оборудования перед Новым годом). Во-вторых, выборка ограничена одним интернет-магазином с относительно небольшим трафиком, что снижает обобщаемость результатов. Наконец, серверная часть не оптимизировалась параллельно с фронтендом, и в условиях пиковых нагрузок именно бэкенд мог становиться узким местом, маскируя потенциально более высокий эффект от обновления интерфейса.

Выводы

Проведенное исследование позволяет сформулировать несколько выводов. Переход от монолитной CMS-архитектуры к компонентному React-приложению с типизацией TypeScript обеспечил снижение времени загрузки страниц каталога и трехкратное уменьшение размера клиентского бандла, что положительно сказалось на пользовательском опыте, особенно на мобильных устройствах.

Адаптивный подход к проектированию интерфейса, основанный не на простом масштабировании десктопного макета, а на перепроектировании информационной архитектуры под разные форм-факторы, позволил повысить конверсию мобильных пользователей и сократить показатель отказов.

Типизация на уровне TypeScript не только снизила число ошибок в ходе разработки, но и обеспечила более прозрачную коммуникацию внутри команды за счет формализации контрактов между компонентами и API.

Результаты работы подтверждают гипотезу о том, что выбор технологического стека фронтенд-разработки может оказывать измеримое влияние на коммерческую эффективность интернет-магазина, и этот фактор заслуживает внимания как разработчиков, так и бизнес-аналитиков при принятии решений о модернизации торговых платформ.

Перспективы дальнейшего исследования связаны с интеграцией серверного рендеринга (SSR) через Next.js для улучшения SEO-показателей, внедрением персонализированных рекомендательных алгоритмов на основе истории просмотров и проведением аналогичного

A/B-тестирования в иных товарных категориях для проверки обобщаемости полученных результатов.

Список литературы:

1. Ешугова С. К., Хамирзова С. К. Развитие электронной коммерции в условиях цифровизации // Новые технологии. 2021. №3. С. 95-104. https://doi.org/10.47370/2072-0920-2021-17-3-95-104

2. Аркабаев Н. К., Алымова З. Ж. Разработка web серверных приложений на базе. NET Core в примере интернет-магазина // Вестник Ошского государственного университета. 2024. №1. С. 142-154. https://doi.org/10.52754/16948610_2024_1_13

3. Потовиченко М. А., Шатилов Ю. Ю. Разработка клиентской части одностраничного web-приложения с использованием библиотеки React // Научное обозрение. Технические науки. 2020. №1. С. 39-43. https://doi.org/10.52754/16948610_2024_1_13

4. Яблокова А. А. Инструменты разработки веб-ориентированных информационных систем при индивидуальном обучении // Информатика. Экономика. Управление-Informatics. Economics. Management. 2023. Т. 2. №2. С. 0214-0223. https://doi.org/10.47813/2782-5280-2023-2-2-0214-0223

5. Омаралиев А., Карабаев С., Омаралиева Г., Вэньхао Д. Методология тестирования безопасности веб-приложений на django с акцентом на выявление уязвимостей бизнес-логики // Вестник Ошского государственного университета. 2025. №4. С. 199-211. https://doi.org/10.52754/16948610_2025_4_14

6. Каюков А. И., Галушка И. А., Коваленко Т. А. Проектирование адаптивных пользовательских интерфейсов // Парадигма. 2025. №5-2. С. 145-151.

7. Аркабаев Н. К., Насиридинова М. Ч., Айбек уулу Д. Оптимизация веб-приложений на Java и HTML5/CSS Grid // Бюллетень науки и практики. 2026. Т. 12. №2. С. 142-152. https://doi.org/10.33619/2414-2948/123/15

8. Евдокимов С. Л. Способы повышения конверсии в электронной торговле с помощью UX-ориентированных решений // Вестник науки. 2025. Т. 1. №3 (84). С. 499-510.

9. Варосян К. К. К Вопросу об особенностях создания интернет-магазина в 2023 году // Human Progress. 2023. Т. 9. №2. С. 4. https://doi.org/10.34709/IM.192.4

References:

1. Eshugova, S., & Khamirzova, S. K. (2021). Razvitie elektronnoi kommertsii v usloviyakh tsifrovizatsii. *Novye tekhnologii,* (3), 95-104. (in Russian). https://doi.org/10.47370/2072-0920-2021-17-3-95-104

2. Arkabaev, N. K., & Alymova, Z. Zh. (2024). Razrabotka web servernykh prilozhenii na baze. NET Core v primere internet-magazina. *Vestnik Oshskogo gosudarstvennogo universiteta,* (1), 142-154. (in Russian).

3. Potovichenko, M. A., & Shatilov, Yu. Yu. (2020). Razrabotka klientskoi chasti odnostranichnogo web-prilozheniya s ispol'zovaniem biblioteki React. *Nauchnoe obozrenie. Tekhnicheskie nauki,* (1), 39-43. (in Russian). https://doi.org/10.52754/16948610_2024_1_13

4. Yablokova, A. A. (2023). Instrumenty razrabotki veb-orientirovannykh informatsionnykh sistem pri individual'nom obuchenii. *Informatika. Ekonomika. Upravlenie-Informatics. Economics. Management, 2*(2), 0214-0223. (in Russian). https://doi.org/10.47813/2782-5280-2023-2-2-0214-0223

5. Omaraliev, A., Karabaev, S., Omaralieva, G., & Ven'khao, D. (2025). Metodologiya testirovaniya bezopasnosti veb-prilozhenii na django s aktsentom na vyyavlenie uyazvimostei biznes-

logiki. *Vestnik Oshskogo gosudarstvennogo universiteta,* (4), 199-211. (in Russian). https://doi.org/10.52754/16948610_2025_4_14

6. Kayukov, A. I., Galushka, I. A., & Kovalenko, T. A. (2025). Proektirovanie adaptivnykh pol'zovatel'skikh interfeisov. *Paradigma,* (5-2), 145-151. (in Russian).

7. Arkabaev, N., Nasiridinova, M., & Aibek uulu, D. (2026). Optimization of Web Applications Using Java and HTML5/CSS Grid. *Bulletin of Science and Practice, 12*(2), 142-152. (in Russian). https://doi.org/10.33619/2414-2948/123/15

8. Evdokimov, S. L. (2025). Sposoby povysheniya konversii v elektronnoi torgovle s pomoshch'yu UX-orientirovannykh reshenii. *Vestnik nauki, 1*(3 (84)), 499-510. (in Russian).

9. Varosyan, K. K. (2023). K voprosu ob osobennostyakh sozdaniya internet-magazina v 2023 godu. *Human Progress*, *9*(2), 4. https://doi.org/10.34709/IM.192.4

Поступила в редакцию
24.04.2026 г.

Принята к публикации
30.04.2026 г.

Ссылка для цитирования:

Аркабаев Н. К., Айтматова Г. Д., Аскарова Л. С. Разработка адаптивного веб-интерфейса на основе React.js и TypeScript как фактор повышения эффективности электронной коммерции компьютерной техники // Бюллетень науки и практики. 2026. Т. 12. №7. С. 87-95. https://doi.org/10.33619/2414-2948/128/09

Cite as (APA):

Arkabaev, N., Aitmatova, G., & Askarova, L. (2026). Development of an Adaptive Web Interface Based on React.js and TypeScript as a Factor in Improving the Efficiency of E-Commerce for Computer Equipment. *Bulletin of Science and Practice, 12*(7), 87-95. (in Russian). https://doi.org/10.33619/2414-2948/128/09