

DOI: <https://doi.org/10.23670/IRJ.2023.133.121>

ОСНОВНЫЕ АСПЕКТЫ РАЗРАБОТКИ МИКРОСЕРВИСНОГО ВЕБ-ПРИЛОЖЕНИЯ

Обзор

Нагорный Н.Н.^{1,*}

¹ ORCID : 0000-0002-8624-684X;

¹ Росбанк, Москва, Российская Федерация

* Корреспондирующий автор (itdevelopernew[at]gmail.com)

Аннотация

В статье рассматриваются основные аспекты разработки веб-приложений. Отмечено, что разработка веб-приложений относится к процессу создания программных приложений, к которым можно получить доступ и использовать через веб-браузер на компьютере или мобильном устройстве. Обоснована актуальность исследования основных аспектов разработки веб-приложения на основе микросервисной архитектуры.

Приведен пример разработки двух алгоритмов: алгоритма поиска пользователей через Интернет и алгоритма добавления пользователя с помощью QR-кода, которое в рамках услуг клиентского сервиса полностью удовлетворяет все требования по функционалу. Использование микросервисной архитектуры при разработке приложений является сегодня наиболее популярным подходом. Микросервисная архитектура для каждого сервиса имеет отдельную упаковку, а на разных серверах они хранятся изолированно, при этом дублируя друг от друга при необходимости, что позволяет отдельно работать с каждым функциональным элементом.

Ключевые слова: технологии, разработка, программное обеспечение, веб-приложение.

MAIN ASPECTS OF MICROSERVICE WEB APPLICATION DEVELOPMENT

Review article

Nagorni N.N.^{1,*}

¹ ORCID : 0000-0002-8624-684X;

¹ Rosbank, Moscow, Russian Federation

* Corresponding author (itdevelopernew[at]gmail.com)

Abstract

The article examines the main aspects of web application development. It is noted that web application development refers to the process of creating software applications that can be accessed and used through a web browser on a computer or mobile device. The relevance of the study of the main aspects of web application development based on microservice architecture is substantiated.

An example of the development of two algorithms is given: an algorithm for searching users via the Internet and an algorithm for adding a user using a QR code, which fully satisfies all functionality requirements within the framework of client services. The use of microservice architecture in application development is the most popular approach today. Microservice architecture has a separate package for each service, and they are stored on different servers in isolation, while duplicating each other if necessary, which allows to work with each functional element separately.

Keywords: technology, development, software, web application.

Введение

В основе решения многих задач научно-технического прогресса лежит обработка информации, которая, в свою очередь, зависит от уровня информатизации общества. Обеспечение свободного доступа к информационным ресурсам формируется за счет разработки надежной и развитой сетевой инфраструктуры [9]. Пропускная способность и надежность в ее работе и направлении зависит от выбора оборудования, которое становится все более востребованным и необходимым. В связи с тем, что быстрое развитие технологий стимулирует производителей на разработку и создание все более разнообразных видов веб-приложений с разной конфигурацией в их характеристиках, то современные веб-приложения могут отличаться параметрами [2]. Поэтому при выборе веб-приложений необходимо четко и ясно понимать задачи, для решения которых они будут использоваться. Они применяются в самых разных отраслях, от электронной коммерции и финансов до здравоохранения и образования, и являются неотъемлемой частью современной цифровой инфраструктуры.

Веб-приложение представляет собой программное приложение, которое работает на веб-сервере и к которому можно получить доступ с помощью веб-браузера. В отличие от традиционных настольных приложений, которые устанавливаются локально на компьютер, доступ к веб-приложениям осуществляется через Интернет и не требует установки на устройство пользователя [3].

Для наиболее оптимального выбора веб-приложений с учетом всех их характеристик и производители, и потребители заинтересованы в постоянном обновлении и апробациях различных приложений, платформ, наиболее удовлетворяющих поставленным целям [10].

В настоящее время индустрия R&D (Research and Development) активно развивается, появляются новые методологии разработки ПО и разных видов архитектуры, соответственно, осложняются задачи для разработчиков.

Основными требованиями к выбору веб-приложений становятся: возможность предоставления программного интерфейса, многофункциональность, легкомасштабируемость, гибкость, кроссплатформенность, быстрая изменяемость под задачи пользователей. В связи с этим для разработки оптимального программного обеспечения необходимо выбрать правильную архитектуру, оценить ее преимущества и недостатки, ориентируясь на требования работоспособности и функциональности ПО. Яркими примерами перехода на микросервисную архитектуру можно назвать компании Netflix, Apple, Instagram*, Pinterest и др., которые начали использовать данную архитектуру, чтобы решить проблемы обработки большого количества запросов, обеспечения высокой скорости доступа к данным, обеспечение высокой надежности отказоустойчивости и масштабируемости. Например, Netflix столкнулся со сложностью обработки информации огромного количества клиентов.

Pinterest благодаря микросервисам смог приспособиться к различным уровням трафика.

Instagram* был впервые запущен в 2010 году как простое монолитное приложение, однако сервер оказался не способен обработать большое количество запросов пользователей. Через полгода на основе микросервисной архитектуры у компании не возникало сложностей с трафиком при работе с тремя миллионами постоянных пользователей.

Когда компания Apple запустила проект Siri, микросервисы позволили расширять и адаптировать программу без перерывов в обслуживании пользователей, тем самым внедрив новый функционал.

Научная новизна исследования заключается в том, что на основе проведенного анализа работоспособности микросервисной архитектуры, ее особенностей выделены и описаны ключевые аспекты, требующие тщательной проработки перед применением микросервисного подхода при разработке веб-приложения.

**(Instagram является продуктом компании Meta, деятельность которой запрещена на территории РФ).*

Постановка задачи и методика исследования

Постановка задачи. Для того чтобы разработчики успешно использовали технологии разработки микросервисного веб-приложения в рамках услуг клиентского сервиса, необходимы рекомендации по правильному их применению.

Методика исследования. На начальном этапе исследования применялись следующие методы: поиск, сравнение и анализ данных для определения особенностей разработки клиент – серверной архитектуры веб-приложений, для выявления их сильных и слабых сторон, а также в разработке алгоритмов поиска пользователей через Интернет и добавления пользователя с помощью QR-кода. Для этого этапа в качестве метода исследования целесообразно использовать моделирование.

Результаты исследования

В работах исследователей особое внимание уделяется технологии разработки программных приложений на основе микросервисной архитектуры.

Н.В. Панченко [7], А.В. Понизов, М.А. Серов, Т.А. Галаган [8] к особенностям микроструктуры приложения относят:

- разделение приложения на микросервисы упрощает параллельную разработку различных компонентов, что ускоряет процесс разработки и внедрения новых функций и обновлений;
- микросервисы обычно взаимодействуют друг с другом посредством API, что облегчает интеграцию с другими системами и сервисами и упрощает добавление новых функциональных возможностей и взаимодействие с внешними системами.

И.С. Голубь придерживается аналогичной позиции, считая, что такой подход к разработке микросервисного веб-приложения в рамках услуг клиентского сервиса позволяет разбить приложение на небольшие, независимые сервисы, каждый из которых выполняет свою специфическую функцию, что обеспечивает гибкость и масштабируемость системы, поскольку каждый сервис может функционировать независимо от других [2].

М.С. Качков, П.А. Пахомов, И.А. Горин отмечают важность выбора инструментальных средств для разработки веб-приложения. В частности, к наиболее оптимальным авторы относят:

1. Языки программирования. В зависимости от требований и предпочтений разработчиков целесообразно использовать JavaScript, HTML, CSS, Python, Ruby и т.д. [3];
2. Фреймворки, которые упрощают разработку веб-приложений, предоставляя готовые модули и инструменты, например: React, Angular, Vue.js, Django, Ruby on Rails и Laravel [1];
3. Среды разработки (IDE), которые предоставляют инструменты для создания, отладки и тестирования кода, например: Visual Studio Code, WebStorm, PyCharm, Sublime Text и Atom;
4. Базы данных. К наиболее популярным относятся реляционные базы данных MySQL, PostgreSQL и SQLite. Для NoSQL баз данных можно использовать MongoDB, Firebase и другие;
5. Веб-сервер, среди которых Apache и Nginx являются наиболее распространенными;
6. API и сервисы, использование которых позволяет интегрировать сторонние сервисы и функциональность в ваше веб-приложение. Некоторые популярные API включают Google Maps API, YouTube API, Twitter API и другие;
7. Тестирование и отладка, например, Jest, Selenium, Mocha, Jasmine и другие;
8. Версионный контроль, например Git, который позволяет отслеживать изменения в коде, сотрудничать с другими разработчиками и управлять версиями приложения [4], [6].

Анализ работы С. Магомадова позволил выделить следующие принципы разработки веб-приложения (Progressive Web Application, далее PWA) [4]:

1. Офлайн работа. PWA должно функционировать в офлайн-режиме, что позволяет пользователям продолжать использовать приложение, даже когда они не имеют доступа к Интернету. С этой целью используются технологии кэширования и локального хранения данных;

2. Адаптивный дизайн, то есть способность приложения адаптироваться к разным устройствам и размерам экранов — это обеспечивает удобство использования и оптимальное отображение интерфейса на различных платформах, включая мобильные устройства;

3. Ориентация на прогрессивное улучшение означает, что PWA предоставляет базовый функционал для всех пользователей, независимо от того, какие возможности доступны на их устройствах или браузерах. В дальнейшем пользователям можно предоставлять дополнительные функции;

4. Быстрая загрузка и отзывчивость. Производительность является важным аспектом PWA. Для быстрой загрузки и реагирования на пользовательские действия используются оптимизации, такие как кэширование ресурсов, сжатие данных и асинхронная загрузка;

5. Обеспечение безопасности данных и защиты пользователей. Для решения данной задачи используются соответствующие протоколы и механизмы шифрования для передачи и хранения данных, а также механизмы аутентификации и авторизации;

6. Возможность автоматического и/или ручного обновления с целью обеспечить актуальность функций и исправление ошибок [5].

Таким образом, учитывая данные принципы, можно сделать вывод о том, что создаваемые веб-приложения, основанные на данных принципах, обладают преимуществами как веб-сайтов, так и нативных приложений, способны обеспечивать достаточный пользовательский опыт и высокую функциональность. В целом, разработка микросервисного веб-приложения в рамках услуг клиентского сервиса может обеспечить более гибкую, масштабируемую и отказоустойчивую систему, способную быстро адаптироваться к изменяющимся требованиям и предоставлять более качественные услуги клиентам.

В этой связи следует подробно рассмотреть клиент-серверную архитектуру веб-приложений и примеры разработки алгоритмов.

Клиент-серверная архитектура веб-приложений – одно из важнейших их характеристик. Клиент-серверная архитектура веб-приложений – это сетевая архитектура веб-приложений, которая предполагает разделение по типу работы компьютеров, находящихся в одной сети, на серверы и клиенты.

Клиент-серверная архитектура веб-приложений также известна как модель сетевых вычислений или сеть клиент-сервер, поскольку все запросы и услуги распределяются по сети. Серверы – это выделенные процессы для управления компьютерами или дисковыми накопителями (файловые серверы), принтерами (серверы печати) или сетевым транспортом (сетевые серверы). Клиенты – это ПК или рабочие станции, на которых пользователи запускают веб-приложения.

Архитектура клиент-серверных веб-приложений, в первую очередь, разработана для сред с достаточно большим количеством компьютеров. Клиент – это компьютерная система, которая выполняет доступ к серверу на других компьютерах через сеть. Клиентские веб-приложения обычно управляют частью пользовательского интерфейса приложения, причем клиентский процесс является внешним интерфейсом программы, которую пользователь видит и с каким взаимодействием существует [4].

Клиентский процесс веб-приложений также управляет локальными ресурсами, которыми он обладает, такими как монитор, клавиатура, центральный процессор рабочей станции. Одним из ключевых элементов клиентской части веб-приложений является графический интерфейс пользователя (GUI).

В клиент-серверной архитектуре веб-приложений серверный процесс – это программа, выполняющая запланированные задачи в ответ на запрос клиента. Сервером могут быть и другие находящиеся в сети устройства, а также операционная система.

Двухуровневая архитектура веб-приложений эффективна, когда клиент непосредственно обращается к серверу. Обычно она используется в небольших помещениях (менее 50 пользователей). Здесь интерфейс пользователя размещается в среде рабочего стола пользователя, а службы веб-приложений – на сервере. Обработка информации разделена между средой интерфейса пользователя системы и средой сервера управления базой данных [5].

В трехуровневой архитектуре клиент-сервера веб-приложений используется дополнительное промежуточное программное обеспечение, в рамках которого запрос клиента проходит на сервер через данный промежуточный уровень, а ответ сервера сначала принимается промежуточным программным обеспечением, а затем клиентом.

Промежуточное программное обеспечение сохраняет всю бизнес-логику и логику доступа к данным. Если их существует несколько, то это называется n-уровневой архитектурой. Промежуточное программное обеспечение может быть файловым сервером, сервером сообщений, сервером приложений, монитором обработки транзакций и т.д. Это повышает гибкость и обеспечивает лучшую производительность.

Сегодня почти все веб-приложения и клиент-серверные приложения спроектированы в виде трехуровневой архитектуры или даже n-уровневой, что позволяет распределить нагрузку между различными серверами и увеличить их мобильность и простоту обслуживания.

Разработанное в статье веб-приложение представляет собой клиент двухзвенной клиент-серверной архитектуры (рис. 1) [6].

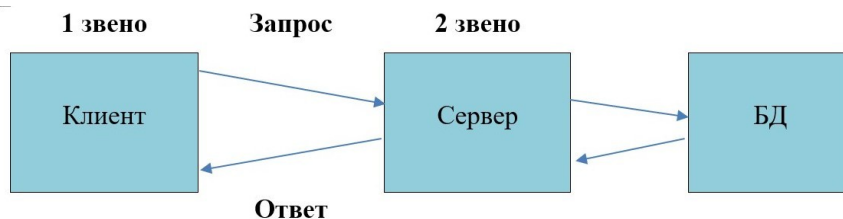


Рисунок 1 - Архитектура программы
DOI: <https://doi.org/10.23670/IRJ.2023.133.121.1>

Клиент - это программное обеспечение, которое предоставляет пользователю интерфейс для взаимодействия с системой. Клиент не взаимодействует с базой данных напрямую и не содержит большого количества бизнес-логики. Клиент отвечает за получение и отображение данных, полученных с сервера, отправку полученных от пользователя данных или выполняет к ним запрос по команде пользователя. В приложении для IOS все взаимодействия с пользователем происходят с помощью сенсорного дисплея. Архитектура IOS программы имеет 4 слоя – рис. 2 [7].

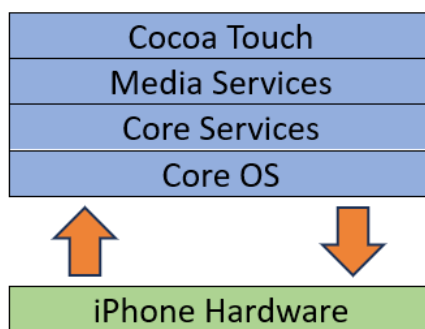


Рисунок 2 - Архитектура IOS программы
DOI: <https://doi.org/10.23670/IRJ.2023.133.121.2>

Рассмотрим пример разработки двух алгоритмов [8]:

1. Алгоритм поиска пользователей через Интернет. Данный алгоритм показывает всю бизнес-логику поиска пользователей через Интернет. В нем представлены три аспекта алгоритма – для Firebase, Backend и самого веб-приложения;

2. Алгоритм добавления пользователя с помощью QR-кода. Разработанный алгоритм показывает всю логику работы веб-приложения для обмена данными через QR-код.

Пример генерации данных через QR-код представлен на рис. 3.

```

- (UIImage *)createNonInterpolatedUIImageFromCIImage:(CIImage *)image
withScale:(CGFloat)scale
{
    // Render the CIImage into a CGImage
    CGImageRef cgImage = [[CIContext contextWithOptions:nil]
createCGImage:image fromRect:image.extent];
    // Now we'll rescale using CoreGraphics
    UIGraphicsBeginImageContext(CGSizeMake(image.extent.size.width * scale,
image.extent.size.width * scale));
    CGContextRef context = UIGraphicsGetCurrentContext();
    // We don't want to interpolate (since we've got a pixel-correct image)
    CGContextSetInterpolationQuality(context, kCGInterpolationNone);
    CGContextDrawImage(context, CGContextGetClipBoundingBox(context),
cgImage);
    // Get the image out
    UIImage *scaledImage = UIGraphicsGetImageFromCurrentImageContext();
    // Tidy up
  
```

Рисунок 3 - Пример отправки данных для воспроизведения звуковых сигналов после кодировки
DOI: <https://doi.org/10.23670/IRJ.2023.133.121.3>

Опишем разработку основных экранов приложения. Разработка программы производилась в среде Xcode, с помощью языка Objective C, по паттерну Apple MVC. Разметка элементов экранов производилась с помощью Constrainit, разметка масштабируется в зависимости от размера экрана с помощью Auto-Layout [9], [10].

Экран приветствия содержит три Label, две Button, одну секцию ScrollView, один UIImageView – рис. 4.

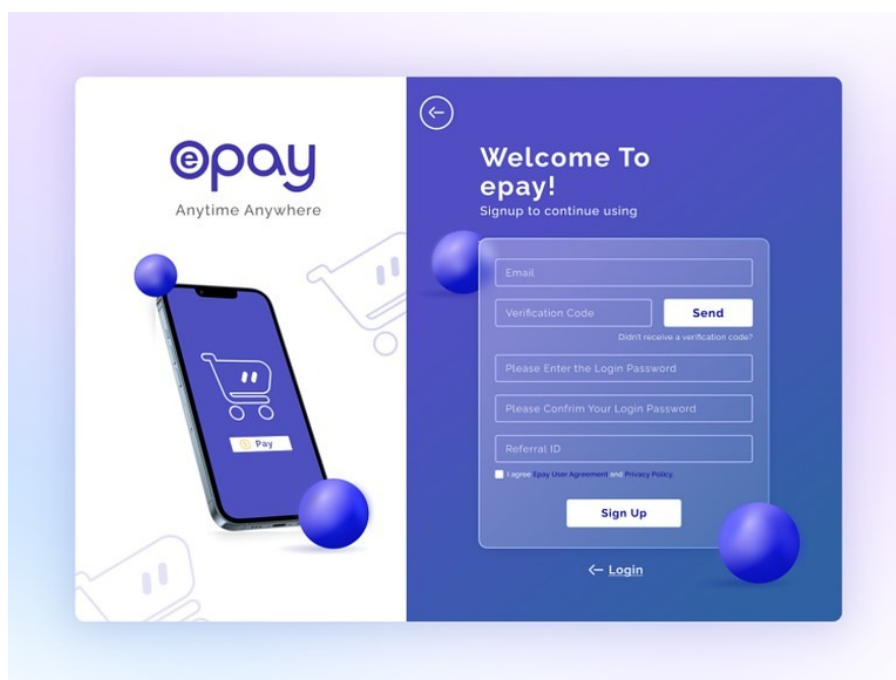


Рисунок 4 - Экран приветствия
DOI: <https://doi.org/10.23670/IRJ.2023.133.121.4>

Экран регистрации позволяет пользователю ввести логин и пароль и подтверждение пароля для создания аккаунта в Firebase.

Экран регистрации содержит два Label, три Button и три TextField – рис. 5.

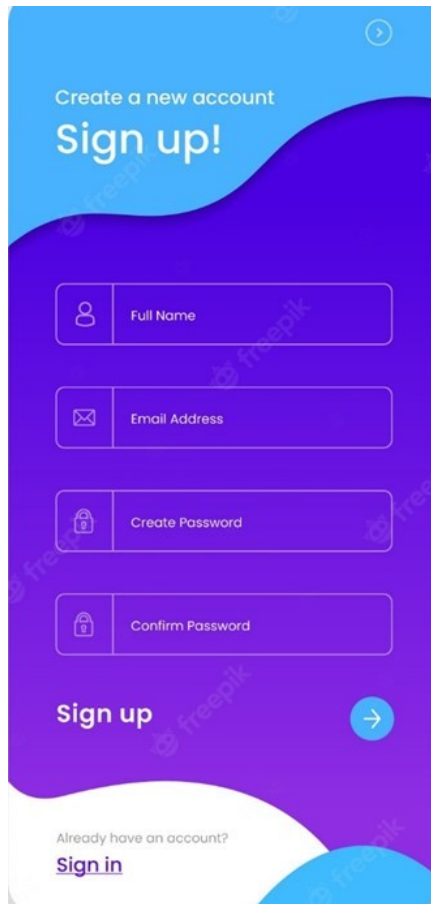


Рисунок 5 - Экран регистрации
DOI: <https://doi.org/10.23670/IRJ.2023.133.121.5>

Экран авторизации позволяет пользователю ввести логин и пароль, используемый в Firebase. Экран авторизации содержит два Label, четыре Button и два TextField – рис. 6.

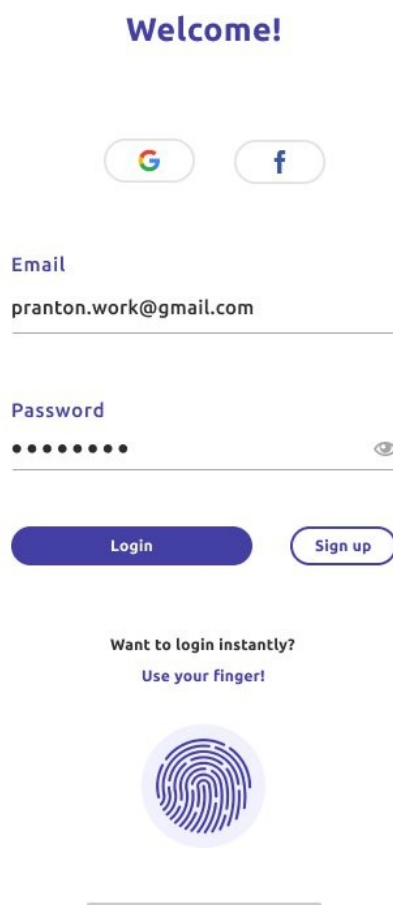


Рисунок 6 - Экран авторизации
DOI: <https://doi.org/10.23670/IRJ.2023.133.121.6>

Заключение

Таким образом, разработка веб-приложения в рамках услуг клиентского сервиса полностью удовлетворяет всем требованиям по функционалу к нему. Использование микросервисной архитектуры при разработке приложений является сегодня наиболее популярным подходом. Микросервисная архитектура для каждого сервиса имеет отдельную упаковку, а на разных серверах они хранятся изолированно, при этом дублируя друг от друга при необходимости, что позволяет отдельно работать с каждым функциональным элементом.

Конфликт интересов

Не указан.

Рецензия

Маняшин А.В., Тюменский Индустриальный университет, Тюмень Российская Федерация
DOI: <https://doi.org/10.23670/IRJ.2023.133.121.7>

Conflict of Interest

None declared.

Review

Manyashin A.V., Tyumen Industrial University, Tyumen Russian Federation
DOI: <https://doi.org/10.23670/IRJ.2023.133.121.7>

Список литературы / References

1. Байдыбеков А.А. Современные фреймворки для разработки веб-приложений / А.А. Байдыбеков, Р.Г. Гильванов, И.А. Молодкин // Интеллектуальные технологии на транспорте. — 2020. — № 4(24). — С. 23-29. — URL: <https://cyberleninka.ru/article/n/sovremennye-freymvorki-dlya-razrabotki-web-prilozheniy> (дата обращения: 26.03.23).
2. Голубь И.С. Компонентный подход в разработке веб приложений / И.С. Голубь // Постулат. — 2020. — № 1(51). — С. 89-92.
3. Елисеева Е.С. Применение JAVASCRIPT-фреймворков при разработке интерактивных образовательных веб-приложений / Е.С. Елисеева, А.Д. Хаханова, А.А. Учанева // Современное образование: традиции и инновации. — 2020. — № 2. — С. 240-243.

4. Качков М.С. Выбор инструментальных средств для разработки образовательного веб-приложения / М.С. Качков, П.А. Пахомов, И.А. Горин // Известия ТулГУ. Технические науки. — 2023. — № 1. — С. 58-62. — URL: <https://cyberleninka.ru/article/n/vybor-instrumentalnyh-sredstv-dlya-razrabotki-obrazovatel'nogo-veb-prilozheniya> (дата обращения: 26.03.23).

5. Магомадов В.С. Основные принципы разработки прогрессивного веб-приложения / В.С. Магомадов // Тенденции развития науки и образования. — 2020. — № 62-4. — С. 81-83.

6. Макеева О.В. Технологии разработки программных приложений / О.В. Макеева, С.А. Красников, М.Б. Туманова [и др.] // Инновации и инвестиции. — 2022. — № 3. — С. 124-127. — URL: <https://cyberleninka.ru/article/n/tehnologii-razrabotki-programmnyh-prilozheniy> (дата обращения: 26.03.23).

7. Панченко Н.В. Особенности разработки веб-приложений и мобильных приложений / Н.В. Панченко // Тенденции развития науки и образования. — 2019. — № 57-2. — С. 24-26.

8. Понизов А.В. Разработка веб-приложения для обработки ГНСС-данных с использованием микросервисной архитектуры / А.В. Понизов, М.А. Серов, Т.А. Галаган // Вестник Амурского государственного университета. Серия: Естественные и экономические науки. — 2020. — № 89. — С. 27-31.

9. Number of smartphone mobile network subscriptions worldwide from 2016 to 2022, with forecasts from 2023 to 2028 // Statista. — 2023. — URL: <https://www.statista.com/statistics/330695> (accessed: 26.03.23).

10. Shamsujjoha M. Human-centric issues in ehealth app development and usage: A preliminary assessment / M. Shamsujjoha, J. Grundy, L. Li [et al.] // 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). — Honolulu, 2021. — P. 506-510. — DOI: 10.1109/SANER50967.2021.00055.

Список литературы на английском языке / References in English

1. Baidybekov A.A. Sovremennye frejmvorki dlya razrabotki web-prilozhenij [Modern frameworks for developing web applications] / A.A. Baidybekov, R.G. Gil'vanov, I.A. Molodkin // Intellektual'nye tekhnologii na transporte [Intelligent technologies in transport]. — 2020. — № 4(24). — P. 23-29. — URL: <https://cyberleninka.ru/article/n/sovremennye-frejmvorki-dlya-razrabotki-web-prilozheniy> (accessed: 26.03.23). [in Russian]

2. Golub I.S. Komponentnyj podhod v razrabotke veb prilozhenij [Component approach in web application development] / I.S. Golub // [Postulat] Postulate. — 2020. — № 1(51). — P. 89-92. [in Russian]

3. Eliseeva E.S. Primenenie JAVASCRIPT-frejmvorkov pri razrabotke interaktivnyh obrazovatel'nyh veb-prilozhenij [Application of JAVASCRIPT frameworks in the development of interactive educational web applications] / E.S. Eliseeva, A.D. Khakhanova, A.A. Uchaneva // Sovremennoe obrazovanie: tradicii i innovacii [Modern education: traditions and innovations]. — 2020. — № 2. — P. 240-243. [in Russian]

4. Kachkov M.S. Vybor instrumental'nyh sredstv dlya razrabotki obrazovatel'nogo veb-prilozheniya [The choice of tools for the development of an educational web application] / M.S. Kachkov, P.A. Pakhomov, I.A. Gorin // Izvestiya TulGU. Tekhnicheskie nauki [Proceedings of TulSU. Technical science]. — 2023. — № 1. — P. 58-62. — URL: <https://cyberleninka.ru/article/n/vybor-instrumentalnyh-sredstv-dlya-razrabotki-obrazovatel'nogo-veb-prilozheniya> (accessed: 26.03.23). [in Russian]

5. Magomadov V.S. Osnovnye principy razrabotki progressivnogo veb-prilozheniya [Basic principles for the development of a progressive web application] / V.S. Magomadov // Tendencii razvitiya nauki i obrazovaniya [Trends in the development of science and education]. — 2020. — № 62-4. — P. 81-83. [in Russian]

6. Makeeva O.V. Tekhnologii razrabotki programmnyh prilozhenij [Technologies for the development of software applications] / O.V. Makeeva, S.A. Krasnikov, M.B. Tumanova [et al.] // Innovacii i investicii [Innovations and investments]. — 2022. — № 3. — P. 124-127. — URL: <https://cyberleninka.ru/article/n/tehnologii-razrabotki-programmnyh-prilozheniy> (accessed: 26.03.23). [in Russian]

7. Panchenko N.V. Osobennosti razrabotki veb-prilozhenij i mobil'nyh prilozhenij [Features of the development of web applications and mobile applications] / N.V. Panchenko // Tendencii razvitiya nauki i obrazovaniya [Trends in the development of science and education]. — 2019. — № 57-2. — P. 24-26. [in Russian]

8. Ponizov A.V. Razrabotka veb-prilozheniya dlya obrabotki GNSS-dannyh s ispol'zovaniem mikroservisnoj arhitektury [Development of a web application for processing GNSS data using microservice architecture] / A.V. Ponizov, M.A. Serov, T.A. Galagan // Vestnik Amurskogo gosudarstvennogo universiteta. Seriya: Estestvennye i ekonomicheskie nauki [Bulletin of the Amur State University. Series: Natural and economic sciences]. — 2020. — № 89. — P. 27-31. [in Russian]

9. Number of smartphone mobile network subscriptions worldwide from 2016 to 2022, with forecasts from 2023 to 2028 // Statista. — 2023. — URL: <https://www.statista.com/statistics/330695> (accessed: 26.03.23).

10. Shamsujjoha M. Human-centric issues in ehealth app development and usage: A preliminary assessment / M. Shamsujjoha, J. Grundy, L. Li [et al.] // 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). — Honolulu, 2021. — P. 506-510. — DOI: 10.1109/SANER50967.2021.00055.