ИНФОРМАТИКА

COMPUTER SCIENCES

Вестник Сыктывкарского университета.
Серия 1: Математика. Механика. Информатика. 2024.
Выпуск 3 (52)
Bulletin of Syktyvkar University.
Series 1: Mathematics. Mechanics. Informatics. 2024; 3 (52)

Научная статья

УДК 05.13.00
https://doi.org/10.34130/1992-2752_2024_3_4

ИНТЕГРАЦИЯ С РАЗЛИЧНЫМИ ПЛАТЕЖНЫМИ СИСТЕМАМИ: ВЫЗОВЫ И РЕШЕНИЯ

Тимур Игоревич Васильев
Paybis LTD Scotland, timmi129@yandex.ru

Аннотация. В современном быстроразвивающемся цифровом мире возможность интеграции с различными платежными системами является наиболее важной для компаний любого размера. Независимо от того, управляете вы небольшим интернет-магазином или же крупной организацией, главным приоритетом всегда должна оставаться возможность предоставления клиентам простой и безопасной оплаты. Однако процесс интеграции с несколькими платежными системами может быть сложным и связан со многими проблемами. Каждая платежная система имеет свой собственный набор требований и тонкостей, поэтому разработчикам и предприятиям важно подходить к интеграции с хорошо продуманной стратегией. В этой статье мы рассмотрим общие проблемы, с которыми сталкиваются при интеграции различных платежных систем, и предложим практические решения для их устранения. К концу этого обсуждения вы получите не только более четкое понимание того, как преодолеть эти сложности, но и сами сможете реализовать надежную интеграцию платежей, отвечающую как потребностям вашего бизнеса, так и ожиданиям ваших клиентов.

Ключевые слова: платежные системы, оптимизация интеграции, автоматизация, веб-приложение

Для цитирования: Васильев Т. И. Интеграция с различными платежными системами: вызовы и решения // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика.* 2024. Вып. 3 (52). С. 4–19. https://doi.org/10.34130/1992-2752_2024_3_4

Article

Integration with various payment systems: challenges and solutions

Timur I. Vasilev
Paybis LTD Scotland, timmi129@yandex.ru

Abstract. In today's fast-paced digital world, the ability to integrate with various payment systems is most important for companies of any size. Whether you run a small online store or a large organization, providing customers with easy and secure payment options should always remain a top priority. However, the process of integrating with multiple payment processors can be complex and present many challenges. Each payment system has its own set of requirements and intricacies, so it is important for developers and businesses to approach integration with a well-thought-out strategy. In this article, we will look at common problems encountered when integrating different payment systems and offer practical solutions to overcome them. By the end of this discussion, you will not only have a clearer understanding of how to overcome these challenges, but you will also be empowered to implement a robust payments integration that meets both your business needs and your customers' expectations.

Keywords: payment systems, integration optimization, automation, web application

For citation: Vasilev T. I. Integration with various payment systems: challenges and solutions. *Vestnik Syktyvkarskogo universiteta. Seriya 1: Matematika. Mekhanika. Informatika* [Bulletin of Syktyvkar University, Series 1: Mathematics. Mechanics. Informatics], 2024, no 3 (52), pp. 4–19. (In Russ.) https://doi.org/10.34130/1992-2752_2024_3_4

1. Введение

В настоящее время для ведения успешного бизнес-проекта необходимо предлагать максимально удобный интерфейс для пользователя, при этом важно учитывать множество аспектов системы, начиная от авторизации и заканчивая удобной оплатой продукта. Независимо от того, какой именно продукт предоставляет компания: банковские услуги, доставка еды или заказ такси, обеспечение корректной работы с платежными системами является самым важным аспектом в ведении бизнеса [1–5].

Чаще всего компании стараются выбирать наиболее быстрый путь, который подразумевает интеграцию с одной конкретной платежной системой. К сожалению, ни один сервис не может в полной мере гарантировать стабильность системы, ведь даже сервисы самых крупных платежных систем вроде Square или Paypal иногда бывают недоступны [6–7]. Такого рода проблема явным образом может сказаться на приложении, и пользователи попросту не смогут оплатить услугу, вследствие чего компания потеряет деньги и лояльность клиентов. Именно поэтому важно интегрироваться с несколькими платежными системами одновременно.

Важно помнить, что любая интеграция с внешним платежным сервисом несет в себе ряд технических сложностей. К примеру, необходимо безопасно хранить ключи взаимодействия, соблюдать юридические нормы, а также обеспечивать стабильную работу интеграции, особенно в условиях высоких нагрузок. При интеграции с несколькими платежными сервисами одновременно появляется множество других проблем, связанных с поддержкой различных API-схем. К одним из таких проблем можно отнести способы авторизации в платежную систему и разный подход к удобству использования этих систем для пользователя. Целью данной работы является анализ и решение самых распространенных проблем, с которыми может столкнуться компания при интеграции с множеством платежных систем одновременно.

2. Материалы и методы

Методы, которыми проводилось исследование: анализ литературы и нормативных документов, сравнение и обобщение опыта интеграции платежных систем, наблюдение, анализ результатов реализации проектов.

При подготовке данной статьи были использованы документации по интеграции следующих платежных систем: ЮKassa, PayPal, BridgerPay, Trustly, Checkout, Square, Stripe [8–14].

3. Результаты

Компаниям перед интеграцией с платёжными системами необходимо решить, существует ли необходимость хранения карточных данных пользователя на их стороне, так как существует строгий стандарт безопасного хранения карт PCI DSS (Payment Card Industry Data Security Standard) [15], которому обязаны следовать все компании, планирующие хранить и обрабатывать данные карт пользователей. Соблюдение требований стандарта PCI DSS может быть затруднительным для компаний из-за большого набора требований к безопасности системы, а также из-за высокой стоимости прохождения ежегодной сертификации. Множество платёжных систем берут обязательства по работе с картами на себя, таким образом, вся банковская информация вводится на стороне платёжной системы, и компаниям нет необходимости соответствовать требованиям PCI DSS.

Первым шагом любой интеграции является изучение API интересующей системы, чаще всего интеграция с платежной системой сводится к нескольким шагам:

- запрос на создание платежа в платежной системе;
- подтверждение оплаты пользователем;
- получение информации об успешной оплате от платежной системы;
- запрос на списание средств в платежную систему.

Звучит достаточно просто, но на деле в данных шагах кроются огромные проблемы, которые необходимо решить.

Первая проблема, с которой придется столкнуться, – это способ аутентификации запросов веб-приложения с платежной системой. На примере интеграции с Яндекс-кассой можно рассмотреть самые популярные из них. Один из распространенных способов — это аутентификация с помощью секретного ключа. Секретный ключ — это уникальная последовательность символов, которая используется для проведения операций. Он необходим, чтобы платежная система могла понять,

что запросы на оплату отправляет авторизованный сервис. Вторым является авторизация с помощью OAuth или JWT, приложению необходимо осуществить запрос с парой логин-пароль, а иногда используя тот же секретный ключ для получения токена, с помощью которого будет осуществляться аутентификация запроса. Например, секретные ключи для аутентификации используются в платежных системах: ЮKassa, Trustly, Checkout; OAuth: ЮKassa, PayPal, Checkout, Square; JWT: BridgerPay. В любом из вышеперечисленных способов аутентификации приложению необходимо безопасно хранить выданный ключ доступа, так как, получив его, злоумышленники смогут осуществлять операции под видом приложения. Для решения данной проблемы стоит использовать различные сервисы безопасного хранения параметров окружения вроде AWS Secrets Manager [16]. С помощью них возможно безопасно хранить параметры окружения и добавлять их в приложение непосредственно во время деплоя.

Также очень часто платежные системы используют дополнительные средства защиты взаимодействия. На примере платежной системы Trustly рассмотрим вызовы API, которые защищены криптографическими подписями. Данная платежная система требует дополнительный уровень безопасности, где каждый API-запрос или вебхук необходимо проверять на корректность криптографической подписи. В данной интеграции используется Base64-хэш всего тела запроса с помощью секретного ключа. Таким образом, компания и Trustly уверены, что запрос является защищенным и осуществляется исключительно между ними.

Исходя из сказанного выше можно сделать вывод о том, что проблемы при использовании нескольких платежных систем могут возникнуть уже на уровне простейшего взаимодействия, ведь необходимо поддерживать несколько видов аутентификации и систем безопасности взаимодействия одновременно, из-за чего разработка может сильно усложниться или замедлиться.

После успешной аутентификации приложение сможет создавать платежи, и в этом случае появляется еще одно различие между платежными системами, а именно идентификатор платежа. Требования к нему отличаются в разных платежных системах, чаще всего системы ограничивают длину идентификатора или запрещают использование сторонних символов, поэтому о корректном идентификаторе необходимо подумать заранее. Очень часто неудачный выбор формата идентификатора может привести к неожиданным сложностям, например, при из-

менении на новый формат вам какое-то время необходимо поддерживать оба формата в системе, так как платёжная система будет ожидать работу по формату, переданному при создании платежа. Лучшей практикой для выбора внешнего идентификатора является использование порядкового номера платежа в системе, таким образом, идентификатор не будет содержать лишних символов и вряд ли превысит максимальный лимит в большинстве платёжных систем.

Следующим моментом, которому следует уделить внимание, является статус платежа в различных интеграциях. Так как в настоящий момент не существует единого формата, которому следуют платежные системы, то нужно проанализировать и разработать свой набор статусов. Также необходимо обезопасить интеграцию и добавить проверку переходов, например, невозможно перевести платеж из статуса «создано» в статус «средства получены», ведь между ними существуют промежуточные статусы, такие как «платеж авторизован» и прочие. Таким образом, система будет защищена от некорректно настроенной интеграции или ошибок в системе. Также всегда следует помнить о том, что платёжная система может отправить некорректные данные или статусы и, добавив такого рода проверку, будет возможно избежать фатальной ошибки и потери средств.

Например, лучше использовать следующий набор статусов в системе:

- Created — платеж создан на стороне системы и ожидает дальнейших действий.
- Cancelled — платеж отменен пользователем.
- Pending — платеж ожидает подтверждения от пользователя (Ввод смс от банка или секретного пароля).
- Declined — произошла ошибка на шаге ожидания подтверждения оплаты пользователем, или пользователь ушел со страницы подтверждения.
- Authorized — подтверждение оплаты пользователем произошло успешно, ожидание дальнейших действий системы, например запуск проверок.
- CaptureRequested — списание средств запрошено системой.

- CaptureFailed — списание средств невозможно.
- Captured — успешное списание средств.
- VoidRequested — аннулирование платежа запрошено системой.
- VoidFailed — аннулирование платежа невозможно.
- Voided — успешное аннулирование платежа.

Таким образом, можно рассмотреть несколько возможных сценариев перехода статусов платежа в системе:

- Пользователь создал платеж, но передумал и отменил заказ. Created => Cancelled
- Пользователь создал платеж, перешел на форму подтверждения платежа и завалил проверку либо решил не проводить платеж. Created => Pending => Declined
- Пользователь создал платеж, прошел проверку от банковской системы, но в результате невозможности оказать услугу было решено отклонить платеж.
  Created => Pending=> Authorized => VoidRequested => Voided
- Пользователь создал платеж, прошел проверку от банковской системы, система запросила списание средств.
  Created => Pending=>Authorized=> CaptureRequested => Captured
- Также стоит упомянуть о вспомогательных статусах CaptureFailed и VoidFailed. Данные статусы необходимы только для критичной ситуации, которая не должна происходить в отлаженной системе, с помощью них можно отследить платежи с некорректными финальными действиями.

Предложенный набор статусов был сформирован на основании анализа следующих платежных систем и их наборов статусов:

- PayPal: Created, Captured, Denied, Voided, Pending;
- Stripe: Canceled, Processing, RequiresAction, RequiresCapture, RequiresConfirmation, Succeeded;

- Square: Approved, Completed, Canceled, Failed;
- ЮKassa: Pending, WaitingForCapture, Succeeded, Canceled.

Статусы платежных систем сильно отличаются друг от друга, и создать набор исключительно из них невозможно. Именно поэтому предложенный выше набор статусов является наиболее подходящим из-за его простоты для понимания и достаточно широкого покрытия всевозможных кейсов при обработке платежа.

Большинство платёжных интеграторов осуществляют оповещение об изменении платёжной операции по средству уведомлений или вебхуков. При построении API для входящих вебхуков рекомендуется придерживаться следующей схемы (рис. 1), то есть все входящие вебхуки от платежной системы передавать на обработку в очередь, например, RabbitMQ или Kafka. Придерживаясь данной архитектурной схемы приложения, можно обработать большое количество входящих запросов при возросшей нагрузке на приложение или при ошибочной отправке большого количества вебхуков от платёжной системы. Также при возникновении внутренней ошибки во время обработки вебхука появится возможность перезапускать обработку сообщений, что обеспечит консистентность данных между системой и платёжным интегратором.

Рис. 1. Схема построения взаимодействия с платежной системой

Многие поставщики платежных услуг предоставляют библиотеки с уже готовым кодом для облегчения интеграции с их системами. Хорошей практикой является использование данных библиотек, так как они поставляются компанией, которая разработала платежную систему. Это позволило им учесть все возможные риски и ошибки, и также

нельзя не упомянуть, что такие библиотеки своевременно обновляются при добавлении новых API или при возникновении проблем с безопасностью [17].

В случае успешной интеграции и решении всех проблем, описанных выше, перед разработчиком неминуемо появится новая дилемма. Необходимо разработать алгоритм по выбору нужной платежной системы, чаще всего этот алгоритм отталкивается от потребностей бизнеса. Некоторые компании используют вспомогательные платежные системы для безопасности, например, если с первой возникнут ошибки или она будет недоступна, компания с легкостью сможет на какое-то время переключиться на другую платежную систему, чаще всего основной выбирают одну из самых приемлемых по тарифу. Также можно разработать алгоритм определения платежной системы по стране карты пользователя или валюты, таким образом существенно оптимизировать неуспешные транзакции по тем или иным причинам.

Помимо технических решений, связанных с бэкендом, частью приложения необходимо проработать удобный UI-флоу пользователя на вебсайте или мобильном приложении. Логика должна работать таким образом, чтобы поддерживать работу с несколькими системами и быть максимально удобной для пользователя. В качестве примера можно рассмотреть процесс оплаты в онлайн-магазине Ozon (рис. 2), после выбора товара пользователь попадает на страницу оформления заказа, где для него предоставляются возможные платёжные методы, и, таким образом, действия по оформлению заказа для пользователя будут всегда аналогичны даже для различных платёжных систем. В статье [18] описаны лучшие практики по созданию удобного интерфейса страницы оплаты.

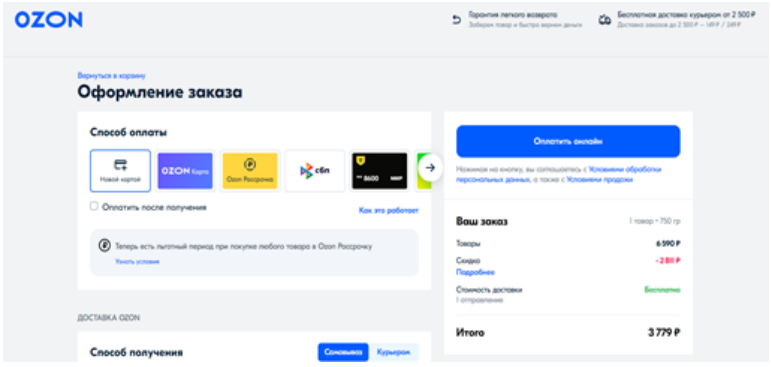

Рис. 2. Пример страницы оплаты товара Ozon

Чтобы добиться удобного UI, необходимо создание единого уровня API на бэкенд части приложения, с помощью которого получится унифицировать процесс. Таким образом, вся логика будет абстрагирована внутри бэкенда, и при добавлении нового платёжного шлюза не придётся дорабатывать ничего, кроме внутренней логики. Для этого желательно придерживаться следующего формата:

- POST public/v1/payments/ — создание платежа.
- GET public/payments/uuid — получение информации о платеже.
- POST internal/v1/payments/uuid/void — аннулирование платежа.
- POST internal/v1/payments/uuid/capture — завершение платежа,

где «internal» префикс обозначает API для внутреннего использования в системе, например для вызова из другого внутреннего сервиса, а «public» префикс – API для использования клиентской частью приложения (мобильное приложение или веб-сайт). Также хорошей практикой для дальнейшего расширения системы является версионирование API с помощью блоков v1/v2/v3 и т. д.

После успешной интеграции с несколькими платежными системами необходимо позаботиться о быстром обнаружении и решении возникающих проблем, для этого потребуется качественная система мониторинга и логирования. Данная система обязана отслеживать логи взаимодействия, статусы платежей, ответы API и данные вебхуков, а также аккумулировать сообщения об ошибках в контексте интеграции со всеми платежными системами. В качестве примера можно привести решение Sentry [19], главной идеей которого является наглядное отображение ошибок в приложении, он имеет удобный дизайн и простую интеграцию с помощью реализованной библиотеки. В качестве системы логирования рекомендуется использовать ELK stack [20], данная система позволяет собирать и обрабатывать огромное количество логов и метрик без негативного воздействия на приложение.

Также не стоит забывать, что интеграция с платежными системами требует надежных методов управления рисками. Различные платежные системы часто имеют собственные инструменты и протоколы управления рисками, которые нужно понимать и качественно с ними работать, если не соблюдать данные требования, ошибки могут привести к финансовым потерям или ухудшению репутации бизнеса.

По мере роста бизнеса компаниям необходимо будет поддерживать повышенную пропускную способность и учитывать проблемы, которые могут возникнуть при масштабировании. Для решения данной проблемы важно распределять запросы между несколькими платежными шлюзами, чтобы минимизировать нагрузку на них, так как это может повлечь дополнительные траты или перегрузить платежный шлюз. Также стоит уделить внимание тем платежным системам, которые на рынке успели зарекомендовать себя, а также предлагают масштабируемость и высокую доступность в любой момент времени [21–22].

4. Обсуждение

Интеграция с несколькими платежными системами — очень сложная, но необходимая задача для компаний, которые хотят обеспечить удобный и безопасный процесс оплаты своим пользователям. Исходя из всего вышеперечисленного можно сделать выводы, что процесс включает в себя решение ряда технических сложностей, связанных с обеспечением поддержки нескольких видов интеграции, управления рисками, удобного пользовательского интерфейса и обеспечения безопасности.

Выводы:

- Необходимо упростить интеграцию с платежными системами и продумать единый API, который облегчит взаимодействие клиентской части с бэкенд-частью приложения.

- Лучше использовать готовые инструменты. Если платежная система выпустила библиотеку с готовым кодом, то необходимо использовать ее при интеграции. Поддержку данного функционала берет на себя компания, выпустившая код, и это заметно скажется на безопасности и скорости разработки внутри системы.

- Внедрение надежной системы мониторинга и логирования ошибок является важным аспектом построения приложения. Разработка функционала приводит к появлению ошибок или проблем, связанных с человеческим фактором. При интеграции с платежной системой необходимо заранее продумать и настроить внутренний мониторинг, чтобы оперативно решать возникшие проблемы.

- При разработке приложения нужно сразу подумать о масштабировании. Необходимо выбирать платежные системы, которые спо-

собны поддерживать тот уровень трафика, что необходим для бизнеса.

- Удобный и интуитивно понятный процесс оплаты является ключом к удержанию клиентов и снижению количества незавершенных покупок. Нужно убедиться, что интерфейс одинаков для различных устройств и работает одинаково для всех интегрированных платежных систем.

Список источников

1. **Chishti S., Craddock T.** The PAYTECH Book // *Wiley*. 2020. Pp. 48–51.

2. **Benson C. C., Loftesness S.** Payment Systems in the U.S.: Third Edition // *Glenbrook Partners*. 2017. Pp. 18–22.

3. **O'Mahony D., Peirce M., Tewari H.** Electronic Payment Systems for E-Commerce // *Artech House*. 2001. Pp. 25–33.

4. **Макембаева К. И., Мырзахматова Ж. Б.** Современные платежные технологии // *Известия вузов Кыргызстана*. 2023. С. 134–138.

5. **Кулумбегова Л. В.** Национальная платежная система России: Оценка состояния и анализ использования платежных инструментов // *Экономика и управление: Проблемы, решения*. 2021. С. 135–140.

6. Описание инцидента сервиса Square. URL: https://developer.squareup.com/blog/incident-summary-2023-09-07 (дата обращения: 22.10.2024).

7. История инцидентов сервиса PayPal. URL: https://www.paypal-status.com/history/production (дата обращения: 22.10.2024).

8. Документация ЮKassa. URL: https://yookassa.ru/developers (дата обращения: 22.10.2024).

9. Документация Paypal. URL: https://developer.paypal.com/docs/online/ (дата обращения: 22.10.2024).

10. Документация BridgerPay. URL: https://developers.bridgerpay.com/docs/getting-started (дата обращения: 22.10.2024).

11. Документация Trsutly. URL: https://amer.developers.trustly.com/payments/docs/overview (дата обращения: 22.10.2024).

12. Документация Checkout. URL: https://api-reference.checkout.com/ (дата обращения: 22.10.2024).

13. Документация Square. URL: https://developer.squareup.com/reference/square/refunds-api (дата обращения: 22.10.2024).

14. Документация Stripe. URL: https://docs.stripe.com/api (дата обращения: 22.10.2024).

15. Описание стандарта PCI DSS. URL: https://selectel.ru/blog/pci-dss/ (дата обращения: 05.09.2024).

16. Документация по работе с AWS Secrets. URL: https://aws.amazon.com/ru/secrets-manager/faqs/ (дата обращения: 05.09.2024).

17. **Pfleeger C. P., Pfleeger S. L., Margulies J.** Security in Computing. 5 ed. Westfield, Massachusetts USA: Prentice Hall, 2015. 910 p.

18. Руководство по созданию страницы оплаты. URL: https://reconcept.ru/blog/kak-sproektirovat-pravil-nyj-interfejs-oplaty (дата обращения: 01.10.2024).

19. Веб-сайт Sentry. URL: https://sentry.io/welcome/ (дата обращения: 01.10.2024).

20. Веб-сайт ELK стека. URL: https://www.elastic.co/elastic-stack (дата обращения: 01.10.2024).

21. **Evans D. S., Schmalensee R.** Paying with Plastic: The Digital Revolution in Buying and Borrowing. 2 ed. Cambridge, Madsachusetts, USA: MIT Press, 2005. 44 p.

22. **Skinner C.** Digital Bank: Strategies to Launch or Become a Digital Bank. Singapore: Marshall Cavendish International Asia Pte Ltd, 2014. 35 p.

References

1. **Chishti S., Craddock T.** The PAYTECH Book. *Wiley*. 2020. Pp. 48–51.

2. **Benson C. C., Loftesness S.** Payment Systems in the U.S.: Third Edition. *Glenbrook Partners*. 2017. Pp. 18–22.

3. **O'Mahony D., Peirce M., Tewari H.** Electronic Payment Systems for E-Commerce. *Artech House*. 2001. Pp. 25–33.

4. **Makembaeva K. I., Myrzakhmatova Zh. B.** Modern payment technologies. *Izvestiya vuzov Kyrgyzstana* [News of universities of Kyrgyzstan]. 2023. Pp. 134–138. (In Russ.)

5. **Kulumbegova L. V.** National payment system of Russia: Assessment of the status and analysis of the use of payment instruments. *Ekonomika i upravlenie: Problemy, resheniya* [Economics and management: Problems, solutions]. 2021. Pp. 135–140. (In Russ.)

6. *Opisanie incidenta servisa Square* [Description of the Square service incident]. Available at: https://developer.squareup.com/blog/incident-summary-2023-09-07 (accessed: 22.10.2024). (In Russ.)

7. *Istoriya incedentov servisa PayPal* [History of PayPal service incidents]. Available at: https://www.paypal-status.com/history/production (accessed: 22.10.2024). (In Russ.)

8. *Dokumentaciya YuKassa* [Yukassa documentation]. Available at: https://yookassa.ru/developers (accessed: 22.10.2024). (In Russ.)

9. *Dokumentaciya Paypal* [Paypal documentation]. Available at: https://developer.paypal.com/docs/online/ (accessed: 22.10.2024). (In Russ.)

10. *Dokumentaciya BridgerPay* [BridgerPay documentation]. Available at: https://developers.bridgerpay.com/docs/getting-started (accessed: 22.10.2024). (In Russ.)

11. *Dokumentaciya Trsutly* [Trsutly documentation]. Available at: https://amer.developers.trustly.com/payments/docs/overview (accessed: 22.10.2024). (In Russ.)

12. *Dokumentaciya Checkout* [Checkout documentation]. Available at: https://api-reference.checkout.com/ (accessed: 22.10.2024). (In Russ.)

13. *Dokumentaciya Square* [Square documentation]. Available at: https://developer.squareup.com/reference/square/refunds-api (accessed: 22.10.2024). (In Russ.)

14. *Dokumentaciya Stripe* [Stripe documentation]. Available at: https://docs.stripe.com/api (accessed: 22.10.2024).

15. *Opisanie standarta PCI DSS* [PCI DSS Description]. Available at: https://selectel.ru/blog/pci-dss/ (accessed: 05.09.2024). (In Russ.)

16. *Dokumentaciya po rabote s AWS Secrets* [Documentation for working with AWS Secrets]. Available at: https://aws.amazon.com/ru/secrets-manager/faqs/ (accessed: 05.09.2024). (In Russ.)

17. **Pfleeger C. P., Pfleeger S. L., Margulies J.** *Security in Computing. 5 ed.* Westfield, Massachusetts USA: Prentice Hall, 2015. 910 p.

18. *Rukovodstvo po sozdaniyu stranicy oplaty* [Guide to creating a payment page]. Available at: https://reconcept.ru/blog/kak-sproektirovat-pravil-nyj-interfejs-oplaty (accessed: 01.10.2024). (In Russ.)

19. *Veb-sajt Sentry* [Sentry web-site]. Available at: https://sentry.io/welcome (accessed: 01.10.2024). (In Russ.)

20. *Veb-sajt ELK steka* [ELK stack web-site]. Available at: https://www.elastic.co/elastic-stack (accessed: 01.10.2024). (In Russ.)

21. **Evans D. S., Schmalensee R.** *Paying with Plastic: The Digital Revolution in Buying and Borrowing.* 2 ed. Cambridge, Madsachusetts, USA: MIT Press, 2005. 44 p.

22. **Skinner C.** *Digital Bank: Strategies to Launch or Become a Digital Bank.* Singapore: Marshall Cavendish International Asia Pte Ltd, 2014. 35 p.

Сведения об авторе / Information about author
Васильев Тимур Игоревич / Timur I. Vasilev
Эксперт в разработке программного обеспечения / Software Development Expert
Ведущий разработчик компании Paybis.com / Senior developer company Paybis.com
g2 1rw, Шотландия, г. Глазго, Уэст-Риджент-стрит, 1 / 1, West egent Street, Glasgow, g2 1rw, Scotland

Статья поступила в редакцию / The article was submitted 21.09.2024
Одобрено после рецензирования / Approved after reviewing 25.10.2024
Принято к публикации / Accepted for publication 27.10.2024