

Реализация интеграции платёжного сервиса ЮKassa для покупки цифровых единиц доступа в веб-приложении

Андриенко Иван Сергеевич

Приамурский государственный университет имени Шолом-Алейхема
Студент

Пащенко Екатерина Александровна

Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация

В статье рассматривается практическая реализация платёжного модуля на базе ЮKassa для приобретения пользователем цифровых единиц доступа, используемых как внутренняя валюта сервиса. Описывается архитектура взаимодействия клиентской и серверной частей, включающая создание платежа на сервере, редирект пользователя на платёжную страницу, последующую проверку статуса операции и начисление цифровых единиц на баланс. Представлена модель хранения платежей в базе данных с фиксацией идентификаторов ЮKassa, статусов, суммы и метаданных, а также обработка вебхуков для автоматического подтверждения оплаты. Дополнительно рассматривается механизм учёта баланса цифровых единиц и журналирование операций начисления и списания с использованием транзакционных операций базы данных. Показано, что предложенный подход обеспечивает отказоустойчивость, безопасность и проверяемость финансовых операций за счёт идемпотентности запросов, серверной валидации параметров и синхронизации статусов с платёжным провайдером.

Ключевые слова: ЮKassa, платёжная интеграция, вебхук, NestJS, TypeORM, транзакции, цифровые кредиты, REST API.

Implementation of the integration of the ЮKassa payment service for the purchase of digital access units in a web application

Andrienko Ivan Sergeevich

Sholom-Aleichem Priamursky State University
Student

Paschenko Ekaterina Alexandrovna

Sholom-Aleichem Priamursky State University
Student

Abstract

The article discusses the practical implementation of a payment module based on ЮKassa for the user to purchase digital access units used as the internal currency of the service. The architecture of the interaction between the client and server parts is described, which includes creating a payment on the server, redirecting the user to the payment page, then checking the transaction status and crediting digital units to the balance. A model for storing payments in a database with the fixation of ЮKassa identifiers, statuses, amounts, and metadata is presented, as well as processing webhooks for automatic payment confirmation. Additionally, the mechanism of accounting for the balance of digital units and logging of accrual and write-off operations using transactional database operations is considered. It is shown that the proposed approach ensures fault tolerance, security and verifiability of financial transactions due to idempotence of requests, server validation of parameters and synchronization of statuses with the payment provider.

Keywords: ЮKassa, payment integration, webhook, NestJS, TypeORM, transactions, digital loans, REST API.

1 Введение**1.1 Актуальность**

В современных веб-сервисах, предоставляющих доступ к функциональности по модели pay-per-use, широко применяется механизм внутренних цифровых единиц, отражающих право пользователя на выполнение ограниченного количества операций. Такая модель требует надёжной интеграции с платёжным провайдером, обеспечивающей корректное создание платежа, безопасный редирект, подтверждение оплаты и неизменность начисления приобретаемых единиц. Существенную роль играет автоматизация подтверждения через вебхуки и хранение состояния платежей в базе данных, поскольку пользователь может завершить оплату не сразу, а внешние системы могут отправлять события с задержкой. Дополнительную сложность вносит необходимость защиты платёжных эндпоинтов, предотвращения повторных начислений, а также ведения прозрачного журнала операций по балансу для последующей проверки. Поэтому разработка модульной архитектуры платёжного контура, сочетающей серверную валидацию, устойчивую синхронизацию статусов и учёт цифровых единиц, является актуальной задачей при построении коммерчески значимых веб-систем.

1.2 Обзор исследований

Перед рассмотрением реализованного механизма оплаты целесообразно опереться на публикации, посвящённые онлайн-платежам и практикам внедрения платёжных модулей в веб-сервисы. В работе Д.А.Казначеевой и С.А. Лузина рассматриваются актуальные альтернативы и особенности рынка онлайн-оплат в условиях кризисного периода 2022 года, что подчёркивает значимость устойчивых и адаптивных платёжных решений для цифровых сервисов [1]. Исследование Л.Н. Акимовой и А.Р. Керимова

описывает практический опыт разработки интернет-магазина, включая организацию пользовательских сценариев покупки и оплат, что демонстрирует прикладную сторону внедрения платёжных механизмов в веб-приложениях [2]. В работе Л.И. Юзвович и А.А. Улановского приведён анализ систем онлайн-платежей, применяемых в электронной торговле, с акцентом на их функциональные возможности и требования к надёжности обработки транзакций [3]. К.Р. Удальцов исследует вопросы обеспечения качества кода в командной разработке, предлагая формализацию шаблонов безопасных pull request-ов для проектов на React и TypeScript в рамках Git Flow [4]. Особое внимание созданию библиотек компонентов на основе TypeScript уделяется в работе Т.М. Воронко, где описываются принципы модульности, повторного использования кода и проектирования архитектуры компонентов для масштабируемых React-решений [5].

1.3 Цель исследования

Целью работы является разработка и описание программного решения по интеграции платёжного сервиса ЮKassa в веб-приложение для покупки цифровых единиц доступа, включая создание платежей на сервере, обработку вебхуков, хранение данных в базе, синхронизацию статусов и корректное начисление единиц на баланс пользователя с журналированием операций.

2 Материалы и методы

Реализация платёжного функционала выполнена на серверной части, построенной на NestJS с использованием TypeORM для работы с базой данных и модульного подхода к организации бизнес-логики. Взаимодействие с ЮKassa API осуществляется через HTTP-клиент axios, а параметры подключения к платёжному провайдеру загружаются через ConfigService из переменных окружения, включая идентификатор магазина, секретный ключ и URL возврата пользователя после оплаты. Для обеспечения идиомпотентности запросов к ЮKassa при создании платежа применяется генерация уникального ключа Idempotence-Key на основе UUID.

3 Результаты и обсуждения

В результате выполненной работы была реализована система онлайн-оплаты, обеспечивающая приобретение цифровых кредитов пользователями веб-приложения. Реализация построена на основе клиент-серверного взаимодействия с использованием платёжного сервиса ЮKassa и ориентирована на автоматическую обработку платежей без участия администратора.

На серверной стороне реализован специализированный сервис, отвечающий за создание платёжных транзакций, их сохранение в базе данных и дальнейшую синхронизацию статусов. Для корректной и безопасной работы используются параметры магазина и секретный ключ, загружаемые из переменных окружения, что позволяет исключить жёсткую привязку конфиденциальных данных к исходному коду (рис. 1).

```
1  import {
2      BadRequestException,
3      Injectable,
4      Logger,
5      UnauthorizedException,
6  } from '@nestjs/common';
7  import { ConfigService } from '@nestjs/config';
8  import { InjectRepository } from '@nestjs/typeorm';
9  import axios from 'axios';
10 import { randomUUID } from 'crypto';
11 import { Repository } from 'typeorm';
12 import { PaymentEntity, PaymentStatus } from '../payment.entity';
13 import { UsersService } from '../users/users.service';
14 import { CreditsService } from '../users/credits.service';
15
16 interface CreatePaymentInput {
17     email: string;
18     credits: number;
19     amount: number;
20     description?: string;
21 }
22
23 @Injectable()
24 export class PaymentsService {
25     private readonly logger = new Logger(PaymentsService.name);
26     private readonly shopId: string;
27     private readonly secretKey: string;
28     private readonly returnUrl: string;
29
30     constructor(
31         private readonly config: ConfigService,
32         @InjectRepository(PaymentEntity)
33         private readonly paymentsRepo: Repository<PaymentEntity>,
34         private readonly usersService: UsersService,
35         private readonly creditsService: CreditsService,
36     ) {
37         this.shopId = this.config.get<string>('YOOKASSA_SHOP_ID') ?? '';
38         this.secretKey = this.config.get<string>('YOOKASSA_SECRET_KEY') ?? '';
39         this.returnUrl =
40             this.config.get<string>('YOOKASSA_RETURN_URL') ??
41             this.config.get<string>('FRONTEND_URL') ??
42             '';
43     }
44 }
```

Рис. 1. Загрузка конфигурации платёжного сервиса и инициализация параметров доступа

Процесс оплаты инициируется созданием записи о платеже в базе данных со статусом *pending*, после чего формируется запрос к API ЮKassa с указанием суммы, валюты, описания платежа и URL возврата пользователя. Дополнительно в запросе передаются метаданные, связывающие транзакцию с конкретным пользователем и количеством приобретаемых кредитов. Полученные от платёжного сервиса идентификатор платежа и ссылка для подтверждения сохраняются на сервере и используются для дальнейшей обработки (рис. 2).

```
45 private assertConfig() {
46   if (!this.shopId || !this.secretKey) {
47     throw new BadRequestException('Yookassa credentials are missing');
48   }
49 }
50
51 async createPayment(input: CreatePaymentInput) {
52   this.assertConfig();
53   const { email, credits, amount, description } = input;
54   if (!email) throw new BadRequestException('email is required');
55   if (!credits || credits <= 0)
56     throw new BadRequestException('credits must be positive');
57   if (!amount || amount <= 0)
58     throw new BadRequestException('amount must be positive');
59
60   const normalizedEmail = `${email}`.trim().toLowerCase();
61   const user = await this.userService.findOrCreate({
62     email: normalizedEmail,
63     name: normalizedEmail,
64   });
65
66   const payment = await this.paymentsRepo.save(
67     this.paymentsRepo.create({
68       status: 'pending',
69       amountValue: amount.toFixed(2),
70       currency: 'RUB',
71       credits,
72       description,
73       userEmail: normalizedEmail,
74       user,
75     }),
76   );
77
78   try {
79     const response = await axios.post(
80       'https://api.yookassa.ru/v3/payments',
81       {
82         amount: { value: amount.toFixed(2), currency: 'RUB' },
83         capture: true,
84         confirmation: {
85           type: 'redirect',
86           return_url: this.returnUrl || 'https://ai-test-generator.ru/profile',
87         },
88         description: description ?? `Покупка ${credits} попыток`,
89         metadata: {
90           paymentId: payment.id,
```

Рис. 2. Формирование запроса на создание платежа и сохранение данных транзакции

Для обеспечения корректного зачисления средств реализован механизм обработки вебхуков, принимающий асинхронные уведомления от платёжной системы. При получении уведомления выполняется проверка подлинности запроса, после чего статус платежа обновляется в базе данных. В случае успешного завершения транзакции осуществляется начисление приобретённых кредитов пользователю, а соответствующая операция фиксируется в журнале изменений баланса (рис. 3).

```
144   async handleWebhook(body: any, headers: Record<string, any>) {
145       this.assertConfig();
146       this.checkWebhookAuth(headers['authorization'] || headers['Authorization']);
147
148       const event = body?.event;
149       const paymentObject = body?.object;
150       if (!paymentObject?.id) {
151           throw new BadRequestException('Invalid webhook payload');
152       }
153
154       const payment = await this.paymentsRepo.findOne({
155           where: { yookassaId: paymentObject.id },
156           relations: ['user'],
157       });
158
159       if (!payment) {
160           this.logger.warn(`Payment not found for webhook id=${paymentObject.id}`);
161           return { ok: true };
162       }
163
164       const newStatus: PaymentStatus =
165           event === 'payment.succeeded'
166               ? 'succeeded'
167               : event === 'payment.canceled'
168                   ? 'canceled'
169                   : payment.status;
170
171       payment.status = newStatus;
172       await this.paymentsRepo.save(payment);
173
174       if (newStatus === 'succeeded') {
175           await this.creditsService.refund(
176               payment.userEmail ?? payment.user?.email ?? '',
177               payment.credits,
178               'payment',
179               {
180                   paymentId: payment.id,
181                   yookassaId: payment.yookassaId,
182                   amount: payment.amountValue,
183               },
184           );
185       }
186
187       return { ok: true };
188   }
```

Рис. 3. Обработка вебхуков платёжной системы и начисление кредитов пользователю

Дополнительно предусмотрен механизм синхронизации статуса платежа по запросу клиента. Если платёж ещё не подтверждён локально, сервер выполняет проверку его состояния через API ЮKassa и, при необходимости, обновляет статус с последующим начислением кредитов. Такой подход позволяет обрабатывать ситуации, при которых вебхук был доставлен с задержкой или не был получен вовсе.

На клиентской стороне реализован интерфейс выбора тарифных планов, обеспечивающий наглядное отображение количества приобретаемых кредитов и их стоимости. После подтверждения оплаты пользователь автоматически перенаправляется на страницу платёжного сервиса, а после

возврата выполняется проверка статуса последнего платежа и обновление пользовательского баланса (рис. 4).

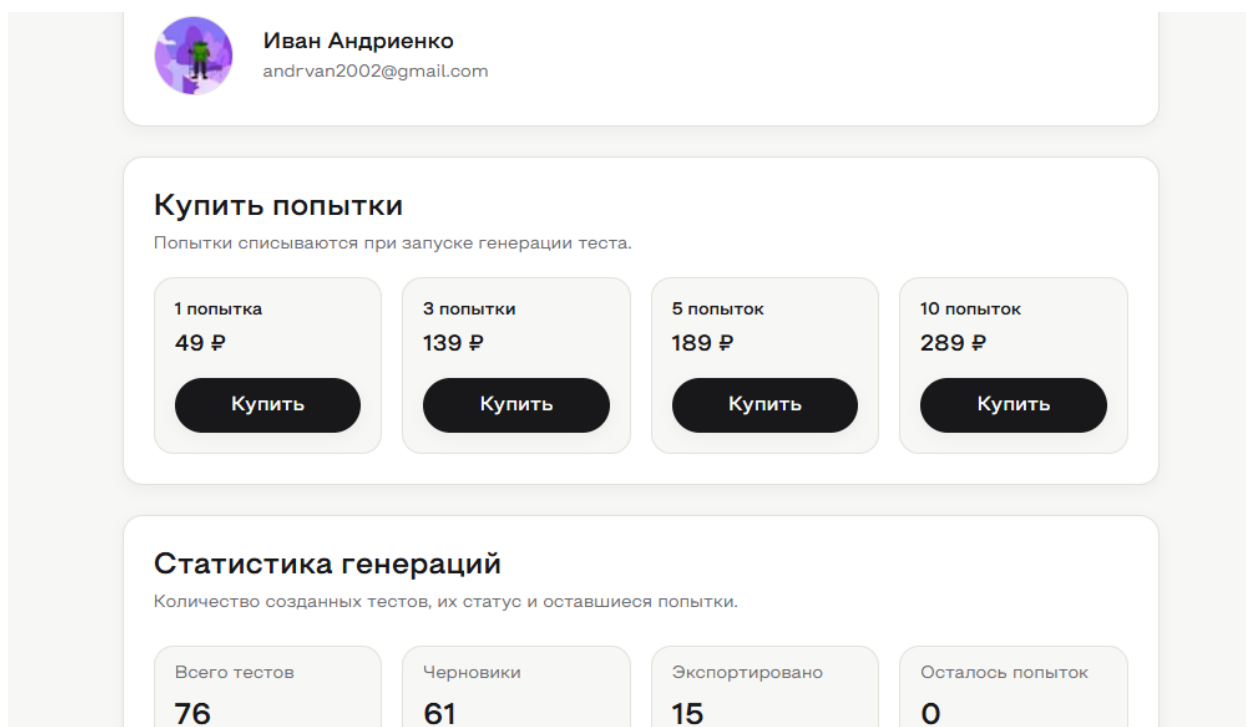


Рис. 4. Пользовательский интерфейс покупки кредитов и проверка статуса платежа после возврата

Реализованная система оплаты обеспечивает надёжное и масштабируемое взаимодействие между пользовательским интерфейсом, серверной логикой и внешним платёжным сервисом. Использование асинхронных уведомлений, журналирования операций и резервной проверки статусов повышает устойчивость системы к сетевым сбоям и обеспечивает корректный учёт финансовых операций.

4 Выводы

В ходе работы была реализована система онлайн-оплаты цифровых услуг с использованием платёжного сервиса ЮKassa, обеспечивающая автоматизированное создание платежей, обработку их статусов и корректное начисление пользовательских кредитов. Реализованное решение демонстрирует надёжную интеграцию клиентской и серверной частей приложения, а также устойчивость к сбоям за счёт использования вебхуков и механизма синхронизации статусов. Полученные результаты подтверждают возможность применения разработанного подхода для построения масштабируемых и безопасных платёжных модулей в веб-приложениях.

Библиографический список

1. Казначеева Д. А., Лузин С. А. Оплата онлайн-покупок и онлайн-сервисов: альтернативы кризис-этапа 2022 года // Студенческая наука: актуальные

- вопросы, достижения и инновации. Сборник статей VIII Международной научно-практической конференции. Пенза, 2022. С. 83–85.
2. Акимова Л. Н., Керимов А. Р. Разработка интернет-магазина “Coffee & Delights” по продаже кофейных напитков // Исследования. Инновации. Практика. 2025. № 3 (16). С. 29–33.
 3. Юзвович Л. И., Улановский А. А. Исследование систем онлайн-платежей, используемых в электронной торговле // Финансы и общество: новые реалии, новые форматы, новые финансовые инструменты. Материалы III Всероссийской научно-практической конференции. Екатеринбург, 2025. С. 40–45.
 4. Свищёва И. В., Посысаев Г. С. Next.js, Nuxt.js и Nest.js: концептуальные и практические различия фреймворков // Инновации. Наука. Образование. 2021. № 46. С. 508–512.
 5. Пензяков П. Д. Возможности фреймворка Nest JS для реализации серверной части веб-систем // Научный альманах Центрального Черноземья. 2022. № 1–4. С. 50–53.