

FILIA
UNIwersYTETU W BIAŁYMSTOKU
W WILNIE
INFORMATYKA



Filia **UwB**
w Wilnie

Daniel Liubianec
Nr albumu: 89413

GAMEVAULT – INTERNETOWA
PLATFORMA SPRZEDAŻY KLUCZY
DO GIER

*Praca dyplomowa napisana
pod kierunkiem
dr. Mariusza Żynela*

Wilno 2026

Składam serdeczne podziękowania Panu dr. Mariuszowi Żynelowi, promotorowi mojej pracy dyplomowej, za poświęcony czas, wsparcie, cierpliwość oraz cenne wskazówki merytoryczne, które pomogły mi w przygotowaniu niniejszej pracy.

Szczególnie dziękuję za zachęcenie mnie do kontynuowania projektu rozpoczętego na drugim roku studiów w ramach przedmiotu *Tworzenie aplikacji webowych* oraz za pomoc w jego dalszym rozwijaniu jako tematu pracy dyplomowej.

Dziękuję również Panu dr. Mieczysławowi Maruszkiewiczowi za życzliwość, inspirujące podejście do studentów oraz pomoc przy realizacji projektu chatbota dla Filii Uniwersytetu w Białymstoku w Wilnie.

Szczególne podziękowania kieruję do Pana Jana Suchodolskiego za pomysł stworzenia chatbota dla Filii, stałe wsparcie, zaangażowanie oraz pomoc na każdym etapie realizacji projektu – od koncepcji aż do jego wdrożenia na stronie internetowej Filii. Dziękuję również za zaproponowanie mi udziału w hackathonie informatycznym, który był dla mnie ciekawym i wartościowym doświadczeniem.

Serdeczne podziękowania składam również Pani dr. Reginie Lašakevič za życzliwość, wsparcie oraz zaproszenia do udziału w wydarzeniach organizowanych na Uniwersytecie. Dziękuję szczególnie za zaproponowanie mi udziału w hackathonie informatycznym, który pozwolił mi zdobyć nowe doświadczenia oraz rozwinąć praktyczne umiejętności informatyczne.

Na zakończenie pragnę również podziękować wszystkim wykładowcom Filii Uniwersytetu w Białymstoku w Wilnie, którzy prowadzili ze mną zajęcia podczas studiów. Dziękuję za przekazaną wiedzę, poświęcony czas oraz wsparcie w rozwijaniu moich umiejętności informatycznych.

Daniel Liubianec

Spis treści

Wstęp	1
1 Kontekst i problem badawczy	2
1.1 Od nośników fizycznych do cyfryzacji	2
1.2 Oficjalne platformy sprzedaży kluczy	3
1.3 Cel i pytania badawcze	6
1.4 Zakres i założenia	6
1.5 Zastosowane technologie	7
1.6 Wkład własny i struktura pracy	8
2 Analiza i projekt systemu	9
2.1 Koncepcja systemu	9
2.2 Role użytkowników	10
2.3 Wymagania funkcjonalne	11
2.4 Wymagania niefunkcjonalne	12
2.5 Diagramy przypadków użycia i BPMN	12
2.6 Model bazy danych	15
3 Implementacja systemu	20
3.1 Autentykacja	20
3.2 Koszyk	25
3.3 Panel administracyjny	28
3.4 Zarządzanie produktami w panelu administratora	30
3.5 Zarządzanie kluczami w panelu administratora	38
3.6 Zarządzanie systemem rabatowym w panelu administratora	42
3.7 Zarządzanie systemem zgłoszeń w panelu administratora	44
3.8 Integracja systemu płatności Stripe	47
4 Testowanie i weryfikacja działania	48
4.1 Weryfikacja standardowego przepływu zakupu	48
4.2 Ocena równoległej prezentacji ofert	54
4.3 Najważniejsze atrybuty karty oferty	56
4.4 Test przejrzystości ceny końcowej	57
4.5 Porównanie mojej platformy z oficjalnymi platformami	57

4.6 Ograniczenia wdrożeniowe projektu	58
Podsumowanie	59
Spis rysunków	61
Spis tabel	63
Bibliografia	64

Wstęp

W ostatnich latach sposób kupowania gier bardzo się zmienił. Zamiast płyt w pudełkach coraz częściej używa się po prostu kluczy cyfrowych albo wykupuje dostęp w formie subskrypcji. Ten sam tytuł może w danej chwili mieć inną cenę na różnych oficjalnych platformach (np. Steam, Epic Games Store), a do tego dochodzą jeszcze takie kwestie jak region aktywacji, wybrana edycja gry, czasowe promocje czy różne zasady zwrotu. Osoba, która chce kupić grę możliwie tanio i na sensownych warunkach, zwykle musi ręcznie przeklikiwać się między kilkoma serwisami i samodzielnie porównywać oferty, co jest po prostu męczące i zajmuje dużo czasu.

W mojej pracy przygotowałem prototyp prostej aplikacji webowej, która zbiera takie informacje w jednym miejscu. Użytkownik widzi obok siebie ceny tej samej gry z kilku platform, a przy każdej pozycji ma od razu oznaczoną platformę, region aktywacji, wariant produktu oraz ostateczną cenę brutto, bez ukrytych opłat.

Na etapie badań korzystałem z danych symulowanych, żeby skupić się głównie na sprawdzeniu samej koncepcji porównywarki i wygody interfejsu. Aplikację napisałem w PHP z wykorzystaniem bazy MySQL (środowisko XAMPP) oraz HTML/CSS i JavaScript. Efektem pracy jest działający prototyp oraz zebrane wnioski na temat jego użyteczności i możliwych kierunków rozbudowy, na przykład o integrację z legalnymi sprzedawcami kluczy cyfrowych.

Rozdział 1

Kontekst i problem badawczy

W ostatniej dekadzie dystrybucja gier i aplikacji wyraźnie przesunęła się od nośników fizycznych ku kluczom cyfrowym oraz modelom subskrypcyjnym [23, 41]. Dla użytkownika oznacza to szybki dostęp do treści, aktualizacje „w locie” i brak logistyki związanej z pudełkami. Jednocześnie ten sam tytuł potrafi w danym momencie kosztować inaczej na różnych oficjalnych platformach (np. Steam i Epic Games Store), a decyzję komplikują dodatkowe atrybuty oferty: region aktywacji, wariant/edycja, czasowa promocja czy polityka zwrotów. W praktyce użytkownik, który chce kupić najkorzystniej, musi porównywać oferty ręcznie – skacząc między platformami. Taka „ręczna” metoda jest czasochłonna i odbiera komfort decyzji. W tym kontekście rośnie potrzeba prostego widoku porównawczego, który zestawia oferty obok siebie i jednoznacznie pokaże kluczowe atrybuty oraz cenę końcową (brutto, bez ukrytych kosztów).

1.1 Od nośników fizycznych do cyfryzacji

Przez lata dominowały nośniki fizyczne: najpierw kasety i dyskietki, później płyty CD/DVD. Model ten miał istotne wady praktyczne. Po pierwsze, awaryjność nośników: płyta lub kaseeta mogła ulec uszkodzeniu (zarysowania, rozmagnesowanie), co prowadziło do utraty danych gry. Po drugie, ograniczona dostępność: nie każdą grę dało się kupić w lokalnym, wyspecjalizowanym sklepie, wiele tytułów docierało z opóźnieniem albo w ograniczonych nakładach. Po trzecie, szczyty popytu: w dniu premiery tworzyły się kolejki trwające od godzin do nawet kilku dni. Zdarzało się, że mimo wizyty w dniu wydania egzemplarze były wyprzedane. Po czwarte, konieczność fizycznej wizyty: zakup wymagał wyjścia z domu, dojazdu i czasu, a aktualizacje/łatki nie były zautomatyzowane.

Upowszechnienie dystrybucji cyfrowej stopniowo zniwelowało te bariery. Około 2003 roku pojawił się Steam, który połączył sklep, bibliotekę i mechanizmy aktualizacji. Wraz z rozwojem ekosystemów zaczęły funkcjonować

klucze cyfrowe przypisujące licencję do konta użytkownika, a później również subskrypcje (stały dostęp do katalogu gier) [44]. W efekcie czas od zakupu do uruchomienia skrócił się do kilku kliknięć, a łatki i aktualizacje stały się procesem w tle. Zmieniła się jednak struktura rynku: pojawiła się regionalizacja ofert, wiele edycji (standard, deluxe, GOTY¹) oraz dynamiczna polityka cenowa.

Współcześnie gracze dzielą się często na dwa obozy. Część użytkowników pozostaje przy pudełkowych wydaniach ze względów kolekcjonerskich, estetycznych lub z myślą o odsprzedaży. Druga grupa preferuje klucze i subskrypcje, stawiając na wygodę, natychmiastowy dostęp i brak logistyki.

1.2 Oficjalne platformy sprzedaży kluczy

Steam – największa platforma dystrybucji cyfrowej na PC. Łączy sklep, bibliotekę gier, aktualizacje oraz funkcje społecznościowe w jednym koncie i aplikacji. Po zakupie gra trafia do biblioteki użytkownika. Wiele tytułów korzysta ze Steam Cloud, który przechowuje zapisy w chmurze i pozwala kontynuować rozgrywkę na innym komputerze [35]. Aktywacja klucza odbywa się oficjalną ścieżką („Activate a Product on Steam”). Po zrealizowaniu kod jest zużyty i nie można go użyć ponownie, a licencja przypina się do konta [31] [36]. Dodatkowo w 2024/2025 roku Valve wprowadziło Steam Families (następca Family Sharing), które upraszczają współdzielenie gier w gospodarstwie domowym. Rozwiązanie to reguluje liczebność rodziny, jednoczesne granie w różne tytuły i zasady opuszczania rodziny [32, 42].

Twórcą Steam jest Gabe Newell „Gaben”, współzałożyciel i wieloletni prezes Valve. Platformę rozwija firma Valve – studio i wydawca znany z serii Half-Life, Portal, Counter-Strike i Dota 2, a także autor silników Source/Source 2 oraz rozwiązań infrastrukturalnych (Steamworks, VAC) [30]. Firma rozwija również sprzęt i technologie: zestaw VR Valve Index oraz przenośny komputer Steam Deck. Te elementy razem oznaczają, że zakup, instalacja, aktualizacje i uruchamianie gier są spójnie powiązane z kontem Steam.



Rysunek 1.1: Logo platformy Steam [43]²

¹Game Of The Year – określenie oznaczające „Grę Roku”. W praktyce termin ten odnosi się również do specjalnych edycji gier (tzw. GOTY Edition), które zawierają podstawową wersję gry oraz dodatkowe treści, takie jak DLC. Przykładem takiej edycji jest Tomb Raider (2013 roku gra): Game of the Year Edition, dostępna na platformie Steam[38].

Epic Games Store – sklep na PC z własnym klientem i biblioteką, w którym gry przypisuje się do konta Epic Games i instaluje przez launcher albo przeglądarkę. Za EGS stoi Epic Games, twórca Unreal Engine 5 – nowoczesnego silnika szeroko używanego przez studia, co wzmacnia pozycję firmy i przyciąga deweloperów. Do szybkiego wzrostu rozpoznawalności sklepu przyczynił się również sukces gry Fortnite, który wprowadził do ekosystemu ogromną liczbę graczy. Kody i karty podarunkowe realizuje się oficjalnie (np. przez stronę <https://epicgames.com/redeem> lub w launcherze), a po ich użyciu treści trafiają do biblioteki użytkownika [9, 7, 8]. Całe doświadczenie zakupowe – podobnie jak w Steam – jest skupione wokół jednego konta i klienta.



Rysunek 1.2: Logo platformy Epic Games [11]

PlayStation Store – cyfrowy sklep dla konsol PS5/PS4. Zakupione gry i dodatki trafiają bezpośrednio do biblioteki użytkownika. Można też zdalnie uruchomić pobieranie z poziomu aplikacji/strony, aby po włączeniu konsoli tytuł był gotowy do startu [25]. Vouchery (12-znakowe kody) realizuje się w interfejsie sklepu. Po poprawnym wpisaniu kodu odpowiedni produkt lub środki portfela przypisują się do konta [28]. Sklep jest ściśle powiązany z usługą PlayStation Plus (plany Essential/Extra/Premium), która zapewnia katalog gier, zapisy w chmurze i rozgrywkę online; wiele tytułów korzysta też z pre-loadu i aktualizacji w tle [27]. Siłą ekosystemu są liczne ekskluzywne serie (np. z katalogu PlayStation Studios takie popularne gry jak The Last Of Us, Spider Man i inne) [26]. Integracja z kontem PSN sprawia, że pobieranie, aktualizacje i dostęp do treści są zautomatyzowane i powiązane z profilem użytkownika.



Rysunek 1.3: Logo platformy PlayStation Store [29]

Microsoft/Xbox – działa na konsolach Xbox oraz w systemie Windows (aplikacja Xbox). Gry i usługi przypisują się do konta Microsoft, a 25-znakowe

²Rysunki lub tabele bez wskazanego źródła bibliograficznego stanowią opracowanie własne autora. W przypadku rysunków pochodzących z zewnętrznych źródeł podano odpowiednie odwołanie bibliograficzne.

kody realizuje się w aplikacjach lub przez stronę [48]. Microsoft podkreśla zalety zakupów cyfrowych: szybki dostęp do biblioteki, dodatków i promocji w jednym miejscu [46]. W ekosystemie PC treści obsługuje aplikacja Xbox, która pozwala instalować gry, zarządzać biblioteką i korzystać z katalogu Game Pass [47]. Taki model minimalizuje operacje po stronie użytkownika – po zakupie gra trafia do konta i może być zdalnie instalowana w środowisku Xbox/Windows.



Rysunek 1.4: Logo platformy Xbox [12]

Marketplaces (Eneba, Kinguin, K4G) – obok oficjalnych sklepów działają rynki wielosprzedawcowe (marketplaces), na których wielu niezależnych sprzedawców wystawia oferty kluczy cyfrowych, a sama platforma zapewnia płatności, panele sprzedawców i często integrację przez API [5].

Eneba określa się jako szybko rosnący marketplace dla graczy, z narzędziami dla sprzedawców (panel, rozliczenia, API) oraz jednym miejscem do porównania wielu źródeł dla kupujących [3, 6].

Kinguin formalnie rozdziela role „użytkownik–sprzedawca” i działa jako administrator serwisu, co potwierdza, że nie jest jednym detalistą wszystkich ofert, lecz platformą łączącą strony transakcji. Regulacje akcentują między innymi obowiązki sprzedawców w zakresie uprawnień do sprzedaży i zgodności z EULA [18, 16].

K4G (Keys for Gamers) również komunikuje model wielosprzedawczy i przyjmuje zewnętrznych dostawców, udostępniając proces wystawiania i wsparcie dla sprzedawców [14, 13].

Dla kupującego marketplaces oznaczają szeroki wybór i konkurencję cenową, ale wymagają uważnego sprawdzania atrybutów oferty (platforma aktywacji, region, wariant/edycja) oraz polityk sprzedawcy. Rynki wielosprzedawcowe w praktyce rozwiązują dwa poważne problemy obecne na oficjalnych platformach, takich jak Steam, Epic Games Store, PlayStation Store i Xbox. Po pierwsze, zamiast „przeskakiwać” między serwisami w celu porównania ofert, gracz może zrobić to w jednym miejscu. Po drugie, oficjalne platformy zwykle trzymają się dość wysokich cen katalogowych, podczas gdy konkurencja wielu sprzedawców na marketplace’ach sprzyja niższym cenom. Z tego powodu wpadłem na pomysł stworzenia własnego, podobnego rozwiązania – prototypu marketplace’u (porównywarki), który w kontrolowanej symulacji adresuje wskazane problemy.

1.3 Cel i pytania badawcze

Celem pracy jest opracowanie i ocena prototypu aplikacji webowej, która prezentuje równoległe oferty dla tego samego tytułu w jednym widoku. Na karcie oferty użytkownik widzi od razu platformę aktywacji, region, wariant/edycję oraz cenę brutto. Badaniu poddaję przede wszystkim użyteczność układu porównawczego i jego wpływ na proces wyboru. Formułuję trzy pytania badawcze:

- (1) Czy równoległa prezentacja ofert skraca czas potrzebny na podjęcie decyzji i zmniejsza ryzyko pomyłki?
- (2) Które atrybuty karty (platforma, region, wariant/edycja) są dla użytkownika najbardziej informacyjne?
- (3) Jak jawna prezentacja ceny końcowej (bez „dokręcania” opłat w kroku koszyka) wpływa na postrzeganą wiarygodność porównania?

1.4 Zakres i założenia

W pracy tworzę prototyp aplikacji webowej, który w jednym miejscu zbiera gry i wybrane aplikacje z różnych ekosystemów oraz zestawia ceny tego samego tytułu „obok siebie” – między platformami i, gdy to sensowne, w ramach tej samej platformy w różnym czasie. Na obecnym etapie prototyp symuluje zebrane oferty i niższe ceny (pokazuje od razu policzoną cenę końcową z podatkami i opłatami, bez konieczności klikania w głąb), aby rzetelnie sprawdzić samą ideę porównania oraz użyteczność interfejsu. W perspektywie rozwojowej planuję włączenie partnerów sprzedających legalne klucze w cenach niższych niż na oficjalnych platformach tak, by porównanie obejmowało zarówno kanały oficjalne, jak i oferty partnerskie.

Zanim przejdę dalej, warto krótko wyjaśnić, czym właściwie jest *klucz cyfrowy*, który jest istotnym pojęciem w mojej pracy. To jednorazowy kod alfanumeryczny, który przypisuje licencję gry lub aplikacji do konta w danej platformie. Aktywacja odbywa się wyłącznie oficjalnymi narzędziami platform (np. „Activate a Product” na Steam, „Redeem Code” w Epic Games). Po aktywacji licencja trafia na konto użytkownika, a klucz staje się zużyty i nie można go użyć ponownie. Obowiązują typowe ograniczenia: region (np. EU-/US), wariant produktu (edycja, DLC³), a czasem także wymagana platforma lub launcher [34]. Proces przypisania licencji jest legalny, bo odbywa się w mechanizmach platformy, jednak w wyjątkowych sytuacjach (np. w przypadku naruszenia regulaminu) licencja może zostać unieważniona [33].

³Downloadable Content – dodatkowa zawartość do gry udostępniana po jej premierze, najczęściej w formie rozszerzeń, nowych misji, elementów fabularnych, map lub przedmiotów w grze [45, 39].

1.5 Zastosowane technologie

XAMPP – to prosty w instalacji pakiet, który stawia na komputerze „mini-serwer” do testów. W środku ma Apache – serwer HTTP, MariaDB – relacyjną bazę danych SQL oraz PHP i Perl. Do tego dochodzi panel sterowania i narzędzia typu phpMyAdmin (prosty panel w przeglądarce do pracy z bazą: podgląd tabel, import i eksport, zapytania). Jednym kliknięciem uruchamia się cały zestaw i można lokalnie odpalać stronę lub API bez żmudnej konfiguracji każdego składnika osobno[1].

PHP – język skryptowy działający po stronie serwera. Kod można wpłacać w HTML. PHP ma bogate biblioteki (obsługa baz, sesji, plików, szyfrowania) i dwa style pracy: proceduralny i obiektowy. Współczesne wersje oferują typowanie, atrybuty i przestrzenie nazw. Na PHP stoi sporo popularnych projektów – od prostych serwisów po aplikacje oparte na frameworkach Laravel czy Symfony [24].

MySQL – relacyjny system bazodanowy oparty na SQL. Daje tabele, indeksy, widoki, procedury składowane oraz transakcje z gwarancjami. W praktyce ceniony jest za szybkość, replikację (kopie danych na innych serwerach) i narzędzia do kopii zapasowych. Dzięki wielu sterownikom łączy się z większością języków programowania [22].

HTML (HyperText Markup Language) – język znaczników, który nadaje stronie strukturę i znaczenie. Opisuje nagłówki, akapity, listy, formularze i multimedia. We współczesnym HTML-u używa się semantycznych tagów (np. `<header>`, `<nav>`, `<main>`, `<article>`), które mówią „co to za część strony”, a nie „jak ma wyglądać”. HTML definiuje strukturę logiczną, inaczej mówiąc „szkielet”, dokumentu, na który nakładamy wygląd (CSS) i zachowanie (JavaScript) [20].

CSS (Cascading Style Sheets) – Arkusze stylów odpowiadają za wygląd dokumentu: kolorystykę, typografię, odstępy i układ elementów. CSS pozwala reagować na kontekst urządzenia poprzez media queries, czyli reguły uruchamiające inne style przy określonej szerokości ekranu, orientacji czy preferencjach systemu (np. tryb ciemny). Dodatkowo zmienne CSS (custom properties, np. `primary-color`) pozwalają trzymać wartości w jednym miejscu i łatwo zmieniać motyw, a animacje oparte na klatkach kluczowych (keyframes) umożliwiają płynne przejścia i ruch bez użycia JavaScript [19].

JavaScript – Jest to język programowania wykonywany po stronie przeglądarki, który odpowiada za logikę i interaktywność aplikacji internetowych. JavaScript umożliwia dynamiczne reagowanie na działania użytkownika, takie jak kliknięcia, wprowadzanie danych czy zmiany stanu formularzy, bez konieczności ponownego ładowania strony[21].

1.6 Wkład własny i struktura pracy

Wkładem własnym jest zaprojektowanie i implementacja prototypu porównywarki cen, w której:

- konsekwentnie oznaczam atrybuty oferty (platforma aktywacji, region, wariant/edycja),
- prezentuję cenę końcową (brutto) wprost na karcie,
- wykorzystuję kontrolowaną symulację danych, aby rzetelnie ocenić interfejs w izolacji od zmienności rynku.

Efektem jest działający demonstrator, który pozwala zweryfikować użyteczność układu „obok siebie” i wyciągnąć wnioski projektowe.

Rozdział 2

Analiza i projekt systemu

W tym rozdziale przedstawiam koncepcję i projekt **GameVault** – łatwej w obsłudze i intuicyjnej platformy pozwalającej na kupowanie kluczy do gier. Z punktu widzenia ról system jest dwuosobowy: użytkownik (klient) i administrator.

Interfejs samej aplikacji podzieliłem na kilka czytelnych widoków: stronę główną z listą i filtrami, kartę produktu (opis, cena, platforma, multimedia), koszyk z podsumowaniem oraz sekcję „Mój profil” z historią kluczy. Osobno działa panel administracyjny z operacjami CRUD – create, read, update i delete. Stąd wynikają wymagania funkcjonalne oraz нефункционалне. Dla uporządkowania zakresu uprawnień dołączam diagramy przypadków użycia (klient i administrator) oraz diagram BPMN procesu zakupu. Również pokażę w jaki sposób aplikacja opiera się na modelu bazy danych.

2.1 Koncepcja systemu

Swoją aplikację webową nazwałem **GameVault**. Jest to lekka aplikacja do zakupu cyfrowych kluczy do gier. Sama nazwa ma dla mnie duże znaczenie – wybrałem ją nieprzypadkowo. Słowo „Vault” rozumiem jako „skarbiec”, czyli jedno miejsce, w którym gromadzone są różne gry i aplikacje na rozmaite platformy.

Zasadnicze założenie w mojej pracy jest proste: w jednym miejscu zebrać ofertę, pozwolić szybko znaleźć tytuł, wybrać wariant (platforma, edycja, region), dodać produkt do koszyka i – po finalizacji – wyświetlić działający klucz zapisany także w historii konta. Moja platforma nie będzie udawała marketplace’u z pośrednikami; stawia na przejrzystość, jednoznaczne atrybuty produktu i krótki proces zakupu.

Przepływ użytkownika w aplikacji jest prosty: strona główna z listą i filtrami, karta produktu (opis, cena brutto, platforma aktywacji, media, trailer), koszyk i podsumowanie, potwierdzenie zakupu oraz wydanie klucza w odpowiednim formacie.

Sam interfejs aplikacji tworzą cztery główne widoki: strona główna, karta produktu, koszyk/zamówienie oraz „Mój profil” z historią zamówień i kluczy. Osobno działa panel administracyjny z operacjami dodawania, edycji i usuwania danych. Najważniejsze akcje („Dodaj do koszyka”, „Przejdź do płatności”, „Zamów”) są zawsze w zasięgu, a komunikaty zwrotne jasno informują o wyniku operacji.

Integralnym elementem jest system zgłoszeń. Użytkownik może opisać problem z konkretnym produktem, dodać załączniki i śledzić status. Administrator prowadzi rozmowę, zmienia priorytety i – gdy potrzeba – przekazuje sprawę innemu administratorowi. Dzięki temu obsługa po sprzedaży odbywa się w jednym miejscu.

2.2 Role użytkowników

W mojej aplikacji rozróżniane są dwie role: użytkownik (klient) i administrator. Informacja o roli jest zapisana w tabeli `users` w polu `role` (`user`, `admin`) i sprawdzana przy każdym żądaniu wymagającym uprawnień. Dzięki temu ten sam mechanizm sesji obsługuje logowanie obu ról, a same role różnią się wyłącznie zakresem dostępnych akcji i widoków.

Użytkownik (klient) – korzysta z części publicznej. Może założyć konto, zalogować się (również z dostępną opcją „Zapamiętaj mnie”), przeglądać i filtrować gry, dodawać pozycje do koszyka, składać zamówienie. W profilu ma wgląd w swoje podstawowe dane (nazwa, adres e-mail), może przeglądać historię przypisanych kluczy oraz ma możliwość zmiany hasła. W razie problemów tworzy zgłoszenia w module ticketów, który zawsze jest widoczny u klienta na ekranie: wybiera produkt, opisuje kłopot, dodaje zrzuty ekranu jako załączniki i śledzi status. Dostęp do ticketu ma tylko właściciel konta i administrator – inni użytkownicy nie zobaczą cudzych zgłoszeń.

Administrator – odpowiada za utrzymanie oferty i wsparcie. Ma dostęp do osobnego panelu, gdzie dodaje nowe gry, edytuje opisy, ceny, multimedia, ustawia platformę i typ produktu (*single*, *bundle*) oraz ukrywa pozycje, które nie powinny być chwilowo widoczne. Może przeglądać listę użytkowników, nadawać lub odbierać uprawnienia administracyjne oraz usuwać konta (operacje zabezpieczone potwierdzeniem). W module ticketów administrator przypisuje zgłoszenia do siebie lub przekazuje innemu administratorowi, ustala priorytet i zmienia statusy (*open*, *in progress*, *resolved*, *closed*), prowadząc jednocześnie rozmowę z użytkownikiem.

2.3 Wymagania funkcjonalne

ID	Wymaganie funkcyjne	Klient	Administrator
WF-01	Rejestracja konta	+	+
WF-02	Logowanie / Wylogowanie	+	+
WF-03	Funkcja „Zapamiętaj mnie”	+	+
WF-04	Przeglądanie produktów	+	+
WF-05	Filtrowanie i sortowanie produktów	+	+
WF-06	Wyświetlanie szczegółów produktu	+	+
WF-07	Dodawanie produktów do koszyka	+	+
WF-08	Zarządzanie koszykiem	+	+
WF-09	Finalizacja zamówienia (checkout)	+	+
WF-10	Otrzymanie kluczy gier po zakupie	+	+
WF-11	Przeglądanie historii zamówień	+	+
WF-12	Przeglądanie zakupionych kluczy	+	+
WF-13	Aktualizacja profilu / Zmiana hasła	+	+
WF-14	Tworzenie ticketów wsparcia	+	+
WF-15	Powiązanie ticketu z produktem	+	+
WF-16	Dodawanie załączników do ticketu	+	+
WF-17	Odpowiadanie na wiadomości w tickecie	+	+
WF-18	Ponowne otwarcie zamkniętego ticketu	+	+
WF-19	Dostęp do panelu administracyjnego	–	+
WF-20	Dodawanie nowych produktów	–	+
WF-21	Edycja istniejących produktów	–	+
WF-22	Usuwanie produktów	–	+
WF-23	Przeglądanie listy użytkowników	–	+
WF-24	Zmiana roli użytkownika	–	+
WF-25	Usuwanie kont użytkowników	–	+
WF-26	Przeglądanie wszystkich ticketów	–	+
WF-27	Zmiana statusu ticketu	–	+
WF-28	Przypisywanie ticketów do siebie	–	+

Tabela 2.1: Wymagania funkcjonalne systemu GameVault

2.4 Wymagania niefunkcjonalne

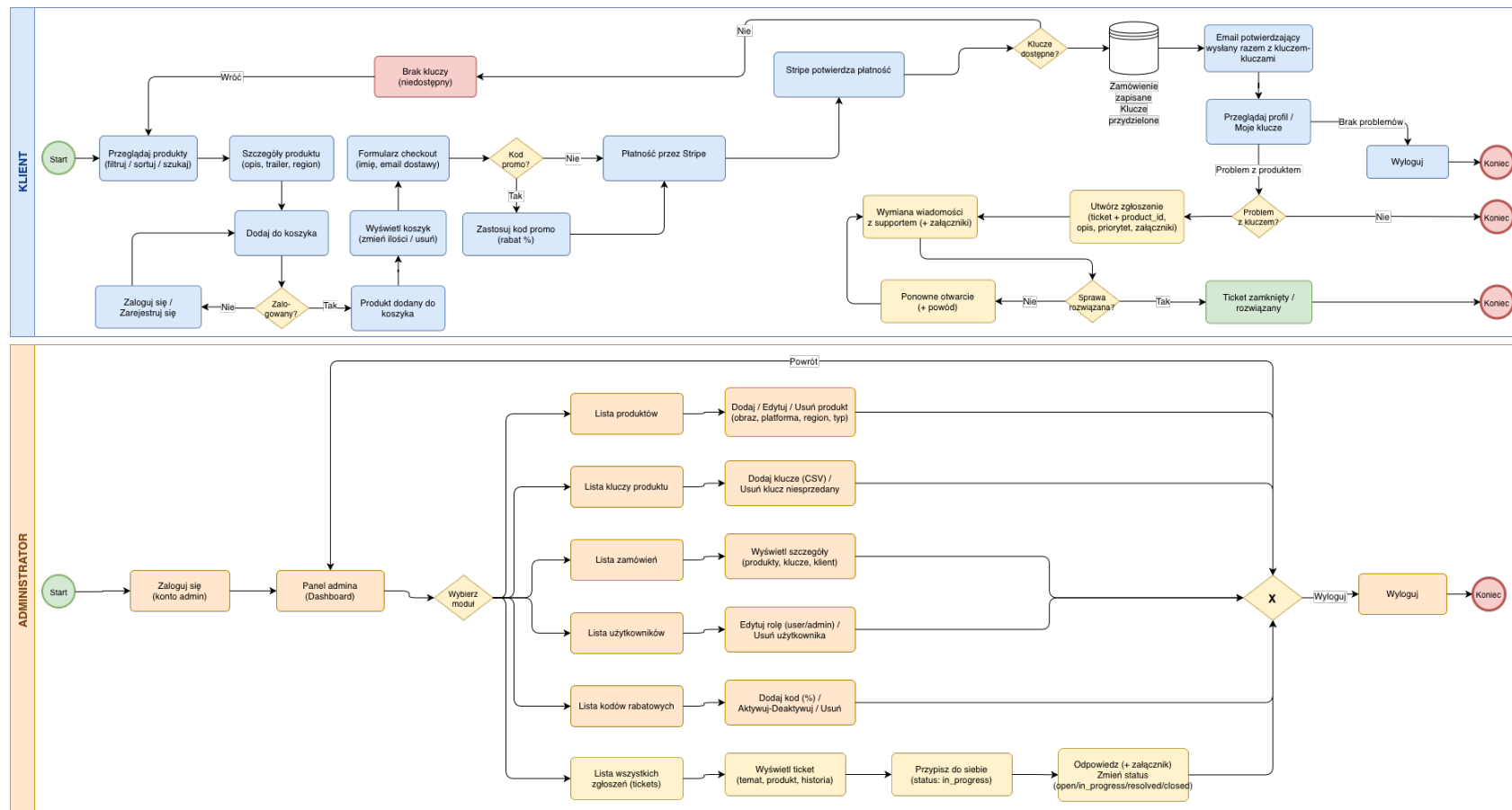
ID	Kategoria	Opis wymagania
WN-01	Bezpieczeństwo	Hashowanie haseł algorytmem bcrypt
WN-02	Bezpieczeństwo	Prepared statements (ochrona przed SQL Injection)
WN-03	Bezpieczeństwo	Escapowanie HTML (ochrona przed XSS)
WN-04	Bezpieczeństwo	Walidacja danych wejściowych po stronie serwera
WN-05	Bezpieczeństwo	Dozwolone typy plików (JPEG, PNG, GIF, WebP)
WN-06	Bezpieczeństwo	Maksymalny rozmiar załączników: 5MB
WN-07	Wydajność	Indeksy na kluczowych kolumnach w bazie danych
WN-08	Wydajność	Transakcje dla operacji krytycznych (checkout)
WN-09	Niezawodność	Integralność referencyjna (klucze obce, CASCADE)
WN-10	Niezawodność	Automatyczne generowanie unikalnych kluczy gier
WN-11	Użyteczność	Responsywny interfejs (RWD)
WN-12	Użyteczność	Interfejs w języku polskim
WN-13	Użyteczność	Komunikaty zwrotne o powodzeniu/błędach
WN-14	Kompatybilność	PHP 8.x, MySQL 8.x
WN-15	Utrzymywalność	Modułowa struktura kodu (separacja plików)

Tabela 2.2: Wymagania niefunkcjonalne systemu GameVault

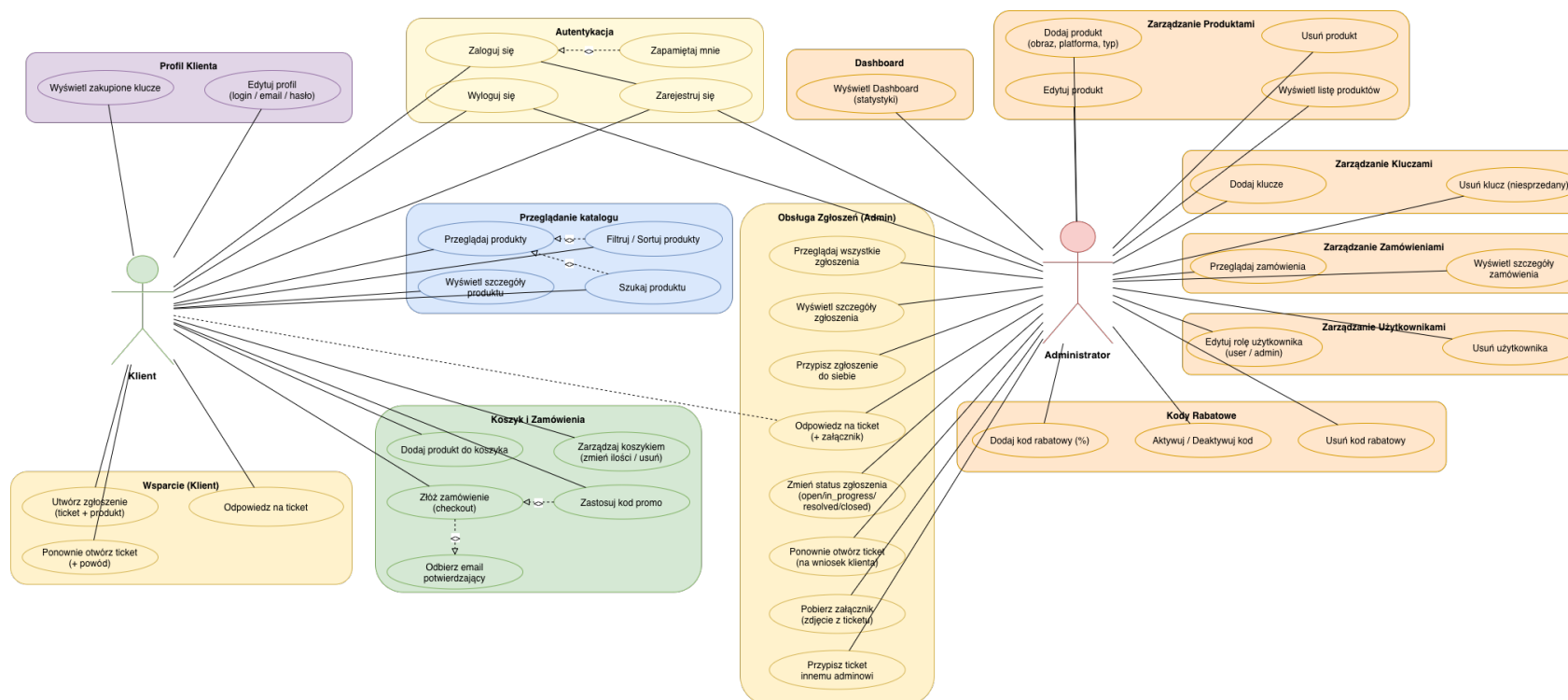
2.5 Diagramy przypadków użycia i BPMN

W tej części pracy chciałbym przedstawić dwa opracowane samodzielnie diagramy. Pierwszym z nich jest diagram BPMN, a drugim diagram przypadków użycia. Celem diagramu BPMN jest zaprezentowanie wybranego procesu. W moim przypadku przedstawia on dwa różne procesy w aplikacji webowej, z perspektywy dwóch ról: klienta oraz administratora. W części dotyczącej klienta zaprezentowałem standardowy przebieg korzystania z aplikacji, natomiast w części dotyczącej administratora przedstawiłem typowy proces zarządzania aplikacją (rys. 2.1).

Kolejnym diagramem jest diagram przypadków użycia. Przedstawia on wszystkie możliwe sposoby wykorzystania aplikacji. Na tym diagramie starałem się uwzględnić wszystkie dostępne działania w systemie, również z podziałem na dwie role: klienta i administratora (rys. 2.2).



Rysunek 2.1: Diagram BPMN



Rysunek 2.2: Diagram przypadków użycia

1. **Moduł użytkowników i produktów** – tabele przechowujące podstawowe dane systemu:
 - `users` – dane użytkowników (klienci i administratorzy),
 - `products` – katalog produktów (gry cyfrowe).
2. **Moduł zakupowy** – tabele obsługujące proces składania zamówień:
 - `cart` – koszyk zakupowy użytkownika,
 - `product_keys` – magazyn kluczy gier przypisanych do produktów (z flagą sprzedaży i informacją o nabywcy),

- **orders** – złożone zamówienia (z obsługą płatności Stripe¹ oraz kodów rabatowych),
- **order_details** – szczegóły zamówień (produkty i ceny),
- **user_keys** – klucze gier przypisane do użytkowników po zakupie,
- **promo_codes** – kody rabatowe z procentową wartością zniżki i flagą aktywności.

3. Moduł wsparcia technicznego – tabele obsługujące system zgłoszeń:

- **tickets** – zgłoszenia użytkowników (z priorytetem, statusem i przypisanym administratorem),
- **ticket_messages** – wiadomości w ramach zgłoszeń,
- **ticket_attachments** – pliki załączone do zgłoszeń.

W tabeli 2.3 znajduje się syntetyczny opis poszczególnych kolumn w bazie danych platformy.

Tabela 2.3: Model bazy danych

Tabela	Kolumna	Opis / Typ danych
— Moduł użytkowników i produktów —		
users	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
	username	VARCHAR(255), UNIQUE, nazwa użytkownika
	email	VARCHAR(255), UNIQUE, adres e-mail
	password	VARCHAR(255), hasło (bcrypt)
	role	ENUM('user', 'admin'), domyślnie 'user'
	created_at	TIMESTAMP, data utworzenia
	remember_token	VARCHAR(255), token sesji
	reset_token	VARCHAR(255), token resetowania hasła
	reset_token_expires	DATETIME, ważność tokenu resetowania
products	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
	name	VARCHAR(255), nazwa produktu
	price	DECIMAL(10,2), cena
	description	TEXT, krótki opis
	detailed_description	TEXT, szczegółowy opis
	trailer_url	VARCHAR(255), URL trailera
	image_data	LONGBLOB, binarne dane obrazu
	mime_type	VARCHAR(100), typ MIME obrazu
	type	ENUM('single', 'bundle'), typ produktu
	platform	VARCHAR(50), platforma (np. Steam)
	region	VARCHAR(50), region sprzedaży
	created_at	TIMESTAMP, data dodania
— Moduł zakupowy —		
cart	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
	user_id	INT, FK → users(id), ON DELETE CASCADE
	product_id	INT, FK → products(id), ON DELETE CASCADE
	quantity	INT, ilość produktów, domyślnie 1

Tabela 2.3

Kontynuacja na następnej stronie

¹Stripe jest platformą płatniczą, która działa jako pośrednik między klientem, sklepem a bankiem.

Tabela 2.3

Kontynuacja z poprzedniej strony

Tabela	Kolumna	Opis / Typ danych
product_keys	<u>created_at</u>	TIMESTAMP, data dodania
	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
	product_id	INT, FK → products(id), ON DELETE CASCADE
	game_key	VARCHAR(100), wartość klucza gry
	region	VARCHAR(50), region klucza
	is_sold	TINYINT(1), flaga sprzedaży, domyślnie 0
	sold_to_user_id	INT, ID użytkownika nabywcy (NULL jeśli niesprzedany)
	sold_at	TIMESTAMP, data sprzedaży (NULL jeśli niesprzedany)
orders	<u>created_at</u>	TIMESTAMP, data dodania
	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
	user_id	INT, FK → users(id), ON DELETE CASCADE
	total	DECIMAL(10,2), łączna wartość zamówienia
	status	ENUM('pending', 'completed', 'failed', 'refunded')
	payment_method	VARCHAR(50), metoda płatności
	promo_code	VARCHAR(50), użyty kod rabatowy
	discount_percent	INT, procentowa wartość zniżki
	promo_code_id	INT, FK → promo_codes(id), ON DELETE SET NULL
	stripe_session_id	VARCHAR(255), identyfikator sesji Stripe
order_details	stripe_payment_intent	VARCHAR(255), identyfikator płatności Stripe
	<u>created_at</u>	TIMESTAMP, data złożenia zamówienia
	<u>updated_at</u>	TIMESTAMP, data ostatniej aktualizacji
	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
user_keys	order_id	INT, FK → orders(id), ON DELETE CASCADE
	product_id	INT, FK → products(id), ON DELETE CASCADE
	quantity	INT, ilość produktów
	price	DECIMAL(10,2), cena jednostkowa
	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
promo_codes	user_id	INT, FK → users(id), ON DELETE CASCADE
	product_id	INT, FK → products(id), ON DELETE CASCADE
	game_key	VARCHAR(50), klucz gry
	region	VARCHAR(50), region klucza
	<u>created_at</u>	TIMESTAMP, data przypisania
	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
ticket_messages	code	VARCHAR(50), UNIQUE, kod rabatowy
	discount_percent	INT, procentowa wartość zniżki
	is_active	TINYINT(1), czy kod jest aktywny, domyślnie 1
	<u>created_at</u>	DATETIME, data utworzenia
— Moduł wsparcia technicznego —		
tickets	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
	user_id	INT, FK → users(id), ON DELETE CASCADE
	product_id	INT, FK → products(id), NULL, ON DELETE SET NULL
	title	VARCHAR(255), tytuł zgłoszenia
	description	TEXT, opis problemu
	status	ENUM('open', 'in_progress', 'resolved', 'closed')
	priority	ENUM('low', 'medium', 'high', 'urgent')
	assigned_admin_id	INT, FK → users(id), NULL, ON DELETE SET NULL
	<u>created_at</u>	TIMESTAMP, data utworzenia
	<u>updated_at</u>	TIMESTAMP, data aktualizacji
ticket_messages	<u>id</u>	INT, klucz główny, AUTO_INCREMENT
	ticket_id	INT, FK → tickets(id), ON DELETE CASCADE
	user_id	INT, FK → users(id), ON DELETE CASCADE

Tabela 2.3

Kontynuacja na następnej stronie

Tabela 2.3

Kontynuacja z poprzedniej strony

Tabela	Kolumna	Opis / Typ danych
ticket_attachments	message	TEXT, treść wiadomości
	is_admin_reply	TINYINT(1), czy odpowiedź admina
	created_at	TIMESTAMP, data wysłania
	id	INT, klucz główny, AUTO_INCREMENT
	ticket_id	INT, FK → tickets(id), ON DELETE CASCADE
	filename	VARCHAR(255), nazwa pliku (systemowa)
	original_filename	VARCHAR(255), oryginalna nazwa pliku
	file_data	LONGBLOB, binarne dane pliku
	file_mime_type	VARCHAR(100), typ MIME pliku (pomocniczy)
	file_size	INT, rozmiar pliku w bajtach
ticket_messages	mime_type	VARCHAR(100), typ MIME pliku
	uploaded_by	INT, FK → users(id), ON DELETE CASCADE
	created_at	TIMESTAMP, data przesłania

Relacje pomiędzy poszczególnymi kolumnami w bazie danych oraz zastosowane klucze obce wraz z akcją `on delete` zostały zebrane w tabeli 2.4.

Tabela	Klucz obcy	Referencja	ON DELETE
cart	user_id	→ users(id)	CASCADE
cart	product_id	→ products(id)	CASCADE
orders	user_id	→ users(id)	CASCADE
orders	promo_code_id	→ promo_codes(id)	SET NULL
order_details	order_id	→ orders(id)	CASCADE
order_details	product_id	→ products(id)	CASCADE
product_keys	product_id	→ products(id)	CASCADE
user_keys	user_id	→ users(id)	CASCADE
user_keys	product_id	→ products(id)	CASCADE
tickets	user_id	→ users(id)	CASCADE
tickets	product_id	→ products(id)	SET NULL
tickets	assigned_admin_id	→ users(id)	SET NULL
ticket_messages	ticket_id	→ tickets(id)	CASCADE
ticket_messages	user_id	→ users(id)	CASCADE
ticket_attachments	ticket_id	→ tickets(id)	CASCADE
ticket_attachments	uploaded_by	→ users(id)	CASCADE

Tabela 2.4: Relacje między tabelami – klucze obce (FK)

Kluczowe cechy projektu bazy danych

- **Integralność referencyjna** – wszystkie relacje między tabelami są zabezpieczone kluczami obcymi (FK) z odpowiednimi akcjami kaskadowymi (`ON DELETE CASCADE` lub `ON DELETE SET NULL`).
- **Indeksowanie** – kluczowe kolumny posiadają indeksy, co przyspiesza wyszukiwanie i łączenie tabel.

- **Typy ENUM** – dla pól o ograniczonym zbiorze wartości (np. role użytkowników: `user`, `admin`, statusy ticketów: `open`, `in_progress`, `resolved`, `closed`, priorytety: `low`, `medium`, `high`, `urgent`) zastosowano typ ENUM, zapewniający walidację na poziomie bazy danych.
- **Automatyczne znaczniki czasu** – większość tabel zawiera kolumnę `created_at` z domyślną wartością `CURRENT_TIMESTAMP`, umożliwiającą śledzenie chronologii operacji.
- **Bezpieczeństwo danych** – hasła przechowywane są w formie zaszyfrowanej algorytmem `bcrypt`, a tokeny sesji (`remember_token`) oraz tokeny resetowania hasła (`reset_token`) przechowywane są jako losowe ciągi znaków o ograniczonym czasie ważności.
- **Integracja z bramką płatności** – w tabeli `orders` przechowywane są identyfikatory sesji i intencji płatności Stripe (`stripe_session_id`, `stripe_payment_intent`), umożliwiając weryfikację i śledzenie transakcji.

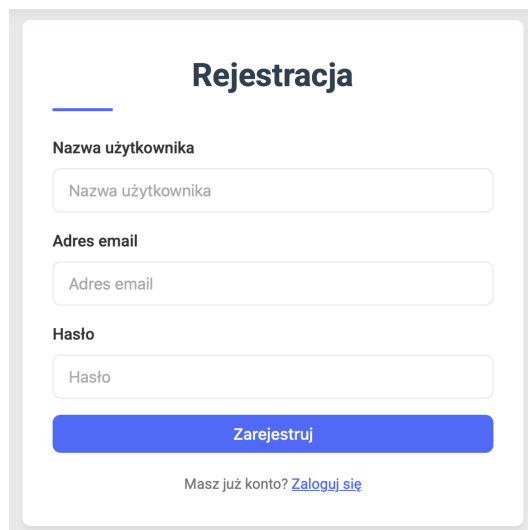
Rozdział 3

Implementacja systemu

W tej części pracy zostaną opisane kluczowe elementy funkcjonalne mojej aplikacji webowej. Do każdego opisu zostaną dołączone istotne fragmenty kodu, a także zrzuty ekranu pokazujące, jak dana funkcja działa i jak wygląda bezpośrednio w aplikacji.

3.1 Autentykacja

Za formularz rejestracji (rys. 3.1) w aplikacji odpowiadają poniższe fragmenty kodu (HTML i PHP). Jego zadaniem jest wypełnienie przez użytkownika trzech pól: nazwę użytkownika, adres e-mail i hasło.

The image shows a web form titled "Rejestracja" (Registration) in bold black text at the top center. Below the title is a horizontal blue line. The form contains three input fields, each with a label above it: "Nazwa użytkownika" (Username), "Adres email" (Email address), and "Hasło" (Password). Each input field has a light gray placeholder text matching its label. Below these fields is a solid blue button with the white text "Zarejestruj" (Register). At the bottom of the form, there is a link that says "Masz już konto? [Zaloguj się](#)" (Do you already have an account? [Log in](#)).

Rysunek 3.1: Formularz rejestracji

Po wysłaniu formularza metodą POST na `index.php?action=register`, serwer najpierw dokonuje sprawdzenia czy podany adres e-mail lub nazwa użytkownika już istnieją w bazie danych (tabela `users`). Jeśli którekolwiek

z nich jest zajęte, wyświetlany jest odpowiedni komunikat błędu. Jeśli dane są unikalne, hasło jest haszowane algorytmem `bcrypt (password_hash())`, następnie nowy rekord użytkownika jest wstawiany do bazy danych, a użytkownik zostaje przekierowany na stronę logowania z komunikatem o pomyślnej rejestracji.

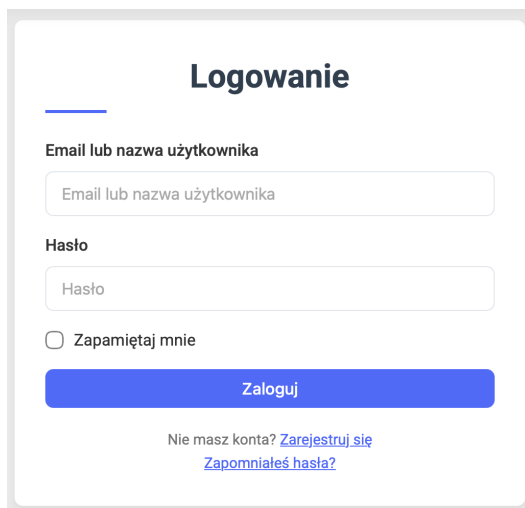
```
case 'register':
    ?>
    <div class="gk-form-container">
        <h2 class="gk-section-title">Rejestracja</h2>
        <?php if (!empty($error)) echo "<p class='gk-error-message'>$error</p>"; ?>
        <form action="index.php?action=register" method="post" class="gk-form">
            <div class="gk-form-group">
                <label for="username" class="gk-form-label">Nazwa użytkownika</label>
                <input type="text" id="username" name="username" class="gk-form-input"
                    placeholder="Nazwa użytkownika"
required>
            </div>
            <div class="gk-form-group">
                <label for="email" class="gk-form-label">Adres email</label>
                <input type="email" id="email" name="email" class="gk-form-input"
                    placeholder="Adres email" required>
            </div>
            <div class="gk-form-group">
                <label for="password" class="gk-form-label">Hasło</label>
                <input type="password" id="password" name="password"
                    class="gk-form-input" placeholder="Hasło"
required>
            </div>
            <button type="submit" class="gk-btn gk-btn-primary
                gk-btn-block">Zarejestruj</button>
        </form>
        <div class="gk-form-footer">
            <p>Masz już konto? <a href="index.php?action=login" class="gk-link">Zaloguj
                się</a></p>
        </div>
    </div>
    <?php
break;
```

```
// Rejestracja
if ($action === 'register' && $_SERVER['REQUEST_METHOD'] === 'POST') {
    $username = $_POST['username'] ?? '';
    $email = $_POST['email'] ?? '';
    $password = $_POST['password'] ?? '';
    $checkEmailSql = "SELECT COUNT(*) AS cnt FROM users WHERE email = ?";
    $checkEmailStmt = $conn->prepare($checkEmailSql);
    $checkEmailStmt->bind_param("s", $email);
    $checkEmailStmt->execute();
    $checkEmailStmt->bind_result($emailCnt);
    $checkEmailStmt->fetch();
    $checkEmailStmt->close();
    $checkUsernameSql = "SELECT COUNT(*) AS cnt FROM users WHERE username = ?";
    $checkUsernameStmt = $conn->prepare($checkUsernameSql);
    $checkUsernameStmt->bind_param("s", $username);
    $checkUsernameStmt->execute();
    $checkUsernameStmt->bind_result($usernameCnt);
    $checkUsernameStmt->fetch();
    $checkUsernameStmt->close();

    if ($emailCnt > 0) {
        $error = "Użytkownik z podanym adresem e-mail już istnieje.";
    }
}
```

```
} elseif ($usernameCnt > 0) {  
    $error = "Nazwa użytkownika jest już zajęta. Wybierz inną nazwę.";  
} else {  
    $passwordHash = password_hash($password, PASSWORD_BCRYPT);  
    $sql = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";  
    $stmt = $conn->prepare($sql);  
    $stmt->bind_param("sss", $username, $email, $passwordHash);  
  
    if ($stmt->execute()) {  
        $stmt->close();  
        header("Location: index.php?action=login&reg_success=1");  
        exit();  
    } else {  
        $error = "Błąd rejestracji: " . $stmt->error;  
    }  
    $stmt->close();  
}  
}
```

Poniższe fragmenty kodu (PHP i HTML) przedstawiają logikę pracy formularza logowania (rys. 3.2), którego celem jest przyjmowanie adresu e-mail lub nazwy użytkownika oraz hasła, z opcjonalnym checkboxem „Zapamiętaj mnie”.



Rysunek 3.2: Formularz logowania

Po wysłaniu formularza metodą POST na `index.php?action=login`, serwer automatycznie rozpoznaje czy podano adres e-mail (walidacja za pomocą funkcji `filter_var()` z parametrem `FILTER_VALIDATE_EMAIL`), czy nazwę użytkownika, a także odpowiednio wyszukuje użytkownika w bazie. Następnym krokiem jest weryfikowanie hasła funkcją `password_verify()` porównującą z haszem `bcrypt`. Jeśli dane po sprawdzeniu są poprawne, ID użytkownika zapisywane jest w sesji. Gdy użytkownik zaznacza „Zapamiętaj mnie”, generowany jest losowy 64-znakowy token (`bin2hex(random_bytes(32))`), później jest zapisywany w ciasteczku (o ważności 30 dni) oraz w bazie danych. Po udanym logowaniu użytkownik jest przekierowywany na stronę swego profilu.

```

case 'login':
?>
<div class="gk-form-container">
  <h2 class="gk-section-title">Logowanie</h2>
  <?php
    if (isset($_GET['reg_success'])) {
      echo "<p class='gk-success-message'>Rejestracja udana, mozesz się teraz
        zalogować.</p>";
    }
    if (!empty($error)) {
      echo "<p class='gk-error-message'>$error</p>";
    }
  ?>
  <form action="index.php?action=login" method="post" class="gk-form">
    <div class="gk-form-group">
      <label for="username" class="gk-form-label">Email lub nazwa
        użytkownika</label>
      <input type="text" id="username" name="username" class="gk-form-input"
        placeholder="Email lub nazwa
użytkownika" required>
    </div>
    <div class="gk-form-group">
      <label for="password" class="gk-form-label">Hasło</label>
      <input type="password" id="password" name="password"
        class="gk-form-input" placeholder="Hasło"
required>
    </div>
    <div class="gk-form-group">
      <div class="gk-checkbox">
        <input type="checkbox" id="remember" name="remember">
        <label for="remember">Zapamiętaj mnie</label>
      </div>
    </div>
    <button type="submit" class="gk-btn gk-btn-primary
      gk-btn-block">Zaloguj</button>
  </form>
  <div class="gk-form-footer">
    <p>Nie masz konta? <a href="index.php?action=register"
      class="gk-link">Zarejestruj się</a></p>
    <p><a href="index.php?action=forgot_password" class="gk-link">Zapomniałeś
      hasła?</a></p>
  </div>
</div>
<?php
break;

```

```

// Logowanie - przetwarzanie POST
if ($action === 'login' && $_SERVER['REQUEST_METHOD'] === 'POST') {
  $username = $_POST['username'] ?? '';
  $password = $_POST['password'] ?? '';
  $remember = isset($_POST['remember']) && $_POST['remember'] === 'on';

  if (filter_var($username, FILTER_VALIDATE_EMAIL)) {

    $sql = "SELECT * FROM users WHERE email = ?";
  } else {

    $sql = "SELECT * FROM users WHERE username = ?";
  }

  $stmt = $conn->prepare($sql);
  $stmt->bind_param("s", $username);
  $stmt->execute();
  $result = $stmt->get_result();

```

```

if ($result && $result->num_rows === 1) {
    $user = $result->fetch_assoc();
    if (password_verify($password, $user['password'])) {
        $_SESSION['user_id'] = $user['id'];

        if ($remember) {

            $checkColumnSql = "SHOW COLUMNS FROM users LIKE 'remember_token'";
            $checkColumnResult = $conn->query($checkColumnSql);

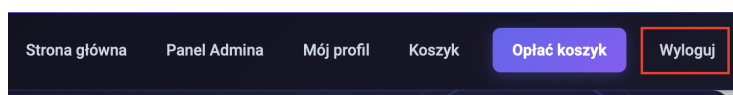
            if ($checkColumnResult && $checkColumnResult->num_rows > 0) {

                $token = bin2hex(random_bytes(32));
                setcookie('remember_token', $token, time() + (30 * 24 * 60 * 60), '/',
                    '', false, true);
                setcookie('user_id', $user['id'], time() + (30 * 24 * 60 * 60), '/',
                    '', false, true);

                $updateTokenSql = "UPDATE users SET remember_token = ? WHERE id = ?";
                $updateTokenStmt = $conn->prepare($updateTokenSql);
                $updateTokenStmt->bind_param("si", $token, $user['id']);
                $updateTokenStmt->execute();
                $updateTokenStmt->close();
            }
        }

        $stmt->close();
        header("Location: index.php?action=profile");
        exit();
    } else {
        $error = "Nieprawidłowe hasło.";
    }
} else {
    $error = "Użytkownik nie został znaleziony.";
}
$stmt->close();
}

```



Rysunek 3.3: Przycisk „Wyloguj”

Po przyciśnięciu przycisku „Wyloguj” (rys. 3.3) z bazy danych usuwany jest token „zapamiętaj mnie” dla aktualnego użytkownika, również kasowane jest ciasteczko z przeglądarki poprzez ustawienie daty wygaśnięcia w przeszłości, a także niszczone jest całe sesja PHP, a użytkownik przekierowywany jest na stronę główną.

```

if ($action === 'logout') {
    if (isset($_SESSION['user_id'])) {
        $sql = "UPDATE users SET remember_token = NULL %WHERE id = ?";
        $stmt = $conn->prepare($sql);
        $stmt->bind_param("i", $_SESSION['user_id']);
        $stmt->execute();
    }
    setcookie('remember_token', '', time() - 3600, "/");
}

```

```
session_destroy();
header("Location: index.php");
exit();
}
```

```
if (!isset($_SESSION['user_id']) && !isset($_COOKIE['remember_token'])) {
    $token = $_COOKIE['remember_token'];
    $hashedToken = hash('sha256', $token);

    $sql = "SELECT id FROM users WHERE remember_token = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("s", $hashedToken);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result && $result->num_rows === 1) {
        $user = $result->fetch_assoc();
        $_SESSION['user_id'] = $user['id'];
    }
}
```

3.2 Koszyk

Za dodawanie produktów do koszyka zakupowego odpowiada poniższy fragment kodu.

```
if ($action === 'add_to_cart' && isset($_GET['product_id'])) {
    if (!isset($_SESSION['user_id'])) {
        exit();
    }

    $userId = $_SESSION['user_id'];
    $productId = (int)$_GET['product_id'];

    $sql = "SELECT quantity FROM cart WHERE user_id = ? AND product_id = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("ii", $userId, $productId);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result && $result->num_rows > 0) {
        $row = $result->fetch_assoc();
        $newQuantity = $row['quantity'] + 1;
        $updateSql = "UPDATE cart SET quantity = ? WHERE user_id = ? AND product_id = ?";
        $updateStmt = $conn->prepare($updateSql);
        $updateStmt->bind_param("iii", $newQuantity, $userId, $productId);
        $updateStmt->execute();
    } else {
        $insertSql = "INSERT INTO cart (user_id, product_id, quantity) VALUES (?, ?, 1)";
        $insertStmt = $conn->prepare($insertSql);
        $insertStmt->bind_param("ii", $userId, $productId);
        $insertStmt->execute();
    }
    header("Location: index.php?action=cart");
    exit();
}
```

Celem danego fragmentu jest w pierwszej kolejności sprawdzenie czy użytkownik jest obecny w sesji. Jeśli użytkownik nie jest zalogowany, to nie może on korzystać z koszyka (rys. 3.4, 3.5).

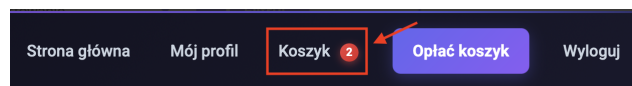


Rysunek 3.4: Dodanie produktu do koszyka poprzez niezalogowaną osobę

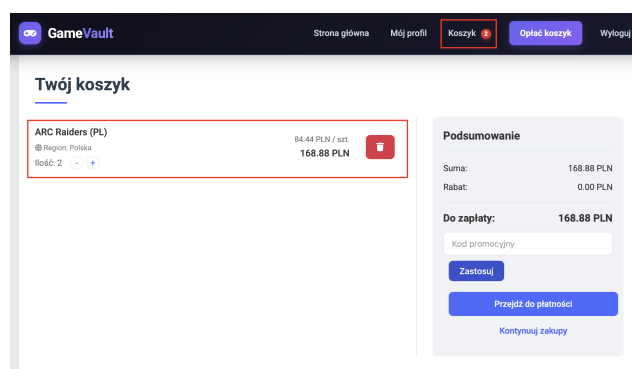


Rysunek 3.5: Komunikat ostrzegawczy jeżeli osoba niezalogowana próbuje dodać produkt do koszyka

Później system dokonuje weryfikacji czy dodany produkt znajduje się w koszyku użytkownika poprzez zapytanie do tabeli `cart`. Jeśli produkt istnieje, wtedy jego ilość zostaje zwiększona o jeden (rys. 3.6). W przeciwnym przypadku tworzony jest nowy rekord w tabeli koszyka z ilością równą jeden. Po zakończeniu operacji użytkownik zostaje przekierowany do widoku koszyka (rys. 3.7).



Rysunek 3.6: Jeśli produkt istnieje w koszyku, to przy ponownym jego dodaniu zwiększa się ilość o jeden

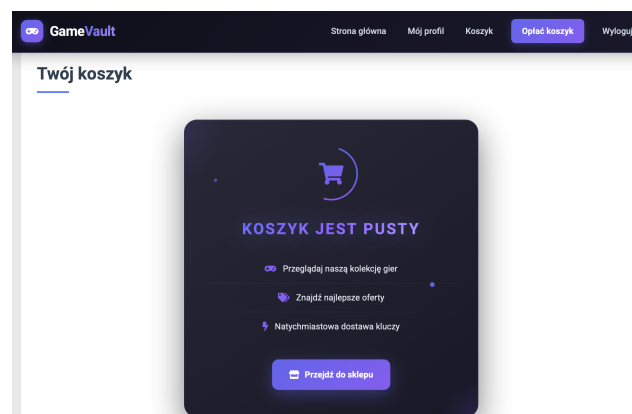


Rysunek 3.7: Przekierowanie do koszyka

Za obsługę usuwania produktów z koszyka zakupowego (rys. 3.8) odpowiada poniższy fragment kodu.

```
if ($action === 'remove_from_cart' && isset($_GET['product_id'])) {  
    if (!isset($_SESSION['user_id'])) {  
        exit();  
    }  
  
    $userId = $_SESSION['user_id'];  
    $productId = (int)$_GET['product_id'];  
  
    $sql = "DELETE FROM cart WHERE user_id = ? AND product_id = ?";  
    $stmt = $conn->prepare($sql);  
    $stmt->bind_param("ii", $userId, $productId);  
    $stmt->execute();  
  
    header("Location: index.php?action=cart");  
    exit();  
}
```

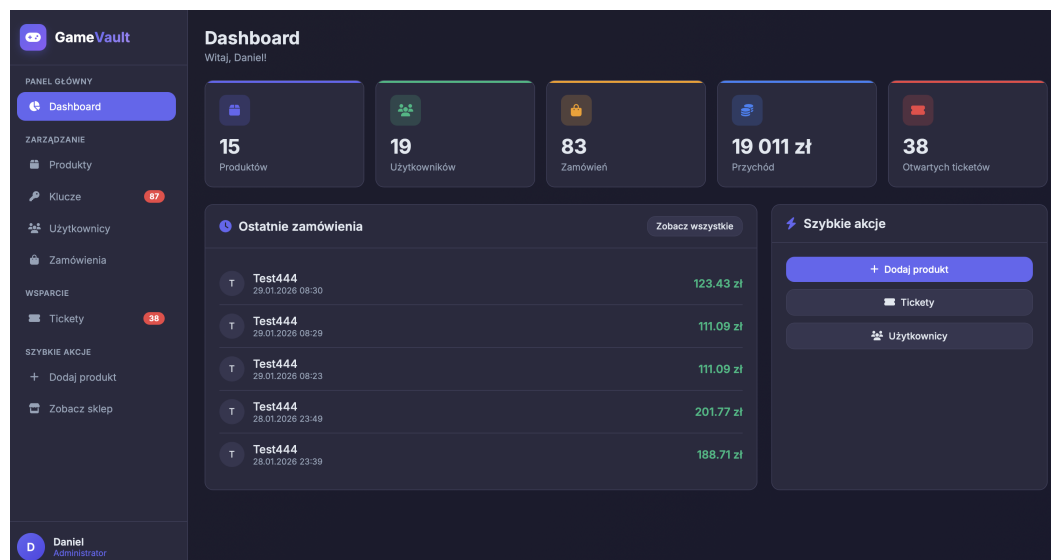
Identyfikator produktu do usunięcia jest pobierany z parametru URL. Następnie wykonywane jest zapytanie DELETE, które dokonuje usunięcia rekordu z tabeli `cart` odpowiadającego kombinacji identyfikatora produktu i użytkownika. Po wykonaniu tej operacji użytkownik jest przekierowywany z powrotem do widoku koszyka.



Rysunek 3.8: Usunięcie produktu z koszyka

3.3 Panel administracyjny

Panel administracyjny wita administratora widokiem Dashboard, który stanowi centralny punkt kontroli nad całą platformą (rys. 3.9).



Rysunek 3.9: Główne okno panelu administracyjnego

W górnej części ekranu prezentowanych jest pięć kart statystycznych ze skrótowym przeglądem stanu sklepu:

- łączna liczba produktów w ofercie,
- zarejestrowanych użytkowników,
- złożonych zamówień,
- wygenerowanego przychodu (w złotych) oraz
- liczba aktualnie otwartych zgłoszeń (ticketów).

Poniżej kart główny obszar dashboardu dzieli się na dwie kolumny:

- po lewej stronie widoczna jest lista pięciu ostatnich zamówień wraz z nazwą klienta, datą i kwotą transakcji, a przycisk „Zobacz wszystkie” przenosi do pełnego widoku zamówień,
- po prawej stronie zgrupowane są Szybkie akcje: „Dodaj produkt”, „Tickety” oraz „Użytkownicy”, umożliwiające natychmiastowe przejście do najczęściej używanych funkcji bez konieczności nawigowania przez boczne menu.

Lewy pasek boczny podzielony jest na cztery sekcje:

- *Panel główny* z linkiem do Dashboardu,
- *Zarządzanie* skupiające dostęp do modułów *Produktów*, *Kluczy* (z licznikiem dostępnych sztuk), *Użytkowników*, *Zamówień* oraz *Kodów rabatowych*,

- *Wsparcie* z modułem Ticketów (oznaczonym licznikiem otwartych zgłoszeń) oraz
- *Szybkie akcje* z bezpośrednimi skrótami do formularza dodawania produktu i podglądu sklepu od strony klienta.

W lewym dolnym rogu paska bocznego wyświetlane są imię i rola aktualnie zalogowanego administratora.

3.4 Zarządzanie produktami w panelu administratora

Kiedy administrator otwiera zakładkę „Produkty” (akcja `action=products`), system wysyła do serwera żądanie wyświetlenia listy produktów (rys. 3.10). Serwer wykonuje zapytanie SQL, które pobiera wszystkie rekordy z tabeli produktów i sortuje je od najnowszego. Dlatego użyto `ORDER BY id DESC`. Następnie wynik zapytania jest przetwarzany w pętli `while`, gdzie każdy produkt jest zamieniany na osobny wiersz w tabeli HTML. W trakcie generowania strony PHP wstawia do kodu dane produktu, takie jak nazwa, cena, region oraz platforma. Aby dane wyświetlane na stronie były bezpieczne, zastosowałem funkcję `htmlspecialchars()`, która zabezpiecza wyjście przed atakiem XSS (czyli próbą wstrzyknięcia złośliwego kodu HTML/JavaScript). Miniatura produktu jest ładowana osobno przez skrypt `get_image.php?id=X`, który pobiera obraz zapisany w bazie danych i zwraca go w formie binarnej.

PRODUKT	REGION	CENA	TYP	AKCJE
 Resident Evil 4 (DLC-Separate Ways) Steam	 Polska	47.12 zł	PAKIET	  
 SnowRunner Xbox	 Łotwa	78.34 zł	KLUCZ	  
 Kingdom Come: Deliverance II Epic Games	 Polska	131.34 zł	KLUCZ	  
 Resident Evil 2 PlayStation	 Europa	124.21 zł	KLUCZ	  
 Cyberpunk 2077 (DLC-Phantom Liberty) Steam	 Litwa	32.41 zł	PAKIET	  
 Hunt: Showdown 1896 (DLC-The Last Laugh) Steam	 Polska	43.21 zł	PAKIET	  
 Battlefield 6 Steam	 Niemcy	143.23 zł	KLUCZ	  

Rysunek 3.10: Lista produktów w zakładce „Produkty” w panelu administratora

```
// PHP - pobieranie i wyświetlanie listy produktów
$products = $conn->query("SELECT * FROM products ORDER BY id DESC");
if ($products && $products->num_rows > 0):
    while ($p = $products->fetch_assoc()):
        ?>
        <tr>
            <td>
                <div class="product-cell">
                    <?php if (!empty($p['image_data'])): ?>
                        
                    <?php endif; ?>
                    <div>
                        <div class="product-name"><?php echo htmlspecialchars($p['name']);
                            ?></div>
                        <div class="product-platform"><?php echo htmlspecialchars($p['platform']
                            ?? 'Steam'); ?></div>
                    </div>
                </div>
            </td>
            <td><strong><?php echo number_format($p['price'], 2); ?> zł</strong></td>
            <td><span class="badge badge-<?php echo $p['type']; ?>">
                <?php echo $p['type'] === 'bundle' ? 'Pakiet' : 'Klucz'; ?>
            </span></td>
            <td>
                <div class="actions">
                    <a href="admin.php?action=edit_product&id=<?php echo $p['id']; ?>" class="btn
                        btn-secondary btn-sm">
                        <i class="fas fa-edit"></i>
                    </a>
                    <a href="admin.php?action=delete_product&id=<?php echo $p['id']; ?>"
                        class="btn btn-danger btn-sm"
                        onclick="return confirm('Usunąć?');">
                        <i class="fas fa-trash"></i>
                    </a>
                </div>
            </td>
        </tr>
    <?php endwhile; endif; ?>
```

Ponieważ lista produktów może zawierać wiele pozycji, w panelu dodano również prostą wyszukiwarkę, która ułatwia szybkie znalezienie konkretnego rekordu (rys. 3.11).



Rysunek 3.11: Pasek wyszukiwania w zakładce „Produkty” w panelu administratora

Funkcja `quickSearch()` napisana w JavaScript nasłuchuje zdarzenia `input` na polu tekstowym `#searchProducts`, czyli reaguje na każdą zmianę wpisywanego tekstu. Następnie skrypt przechodzi przez wszystkie wiersze tabeli i sprawdza, czy ich pełna treść (`textContent`) zawiera wpisaną frazę. Jeżeli

dany produkt nie pasuje do wyszukiwania, jego wiersz jest ukrywany poprzez `style.display = 'none'`. W sytuacji, gdy żaden wiersz nie spełnia warunku, system dynamicznie wyświetla komunikat „Brak wyników”. Dodatkowo zastosowano skrót klawiaturowy `Ctrl+K` / `Cmd+K`, który automatycznie przenosi fokus na pole wyszukiwania, dzięki czemu użytkownik może od razu zacząć wpisywanie.

```
<!-- HTML - pole wyszukiwania -->
<div class="search-box">
  <i class="fas fa-search search-icon"></i>
  <input type="text" id="searchProducts" class="search-input"
    placeholder="Szukaj produktów po nazwie, platformie...">
</div>
<table class="table" id="productsTable">...</table>

// JavaScript - logika filtrowania wierszy tabeli
function quickSearch(inputId, tableId) {
  const input = document.getElementById(inputId);
  const table = document.getElementById(tableId);

  input.addEventListener('input', function() {
    const searchTerm = this.value.toLowerCase().trim();
    const rows = table.querySelectorAll('tbody tr');
    let visibleCount = 0;

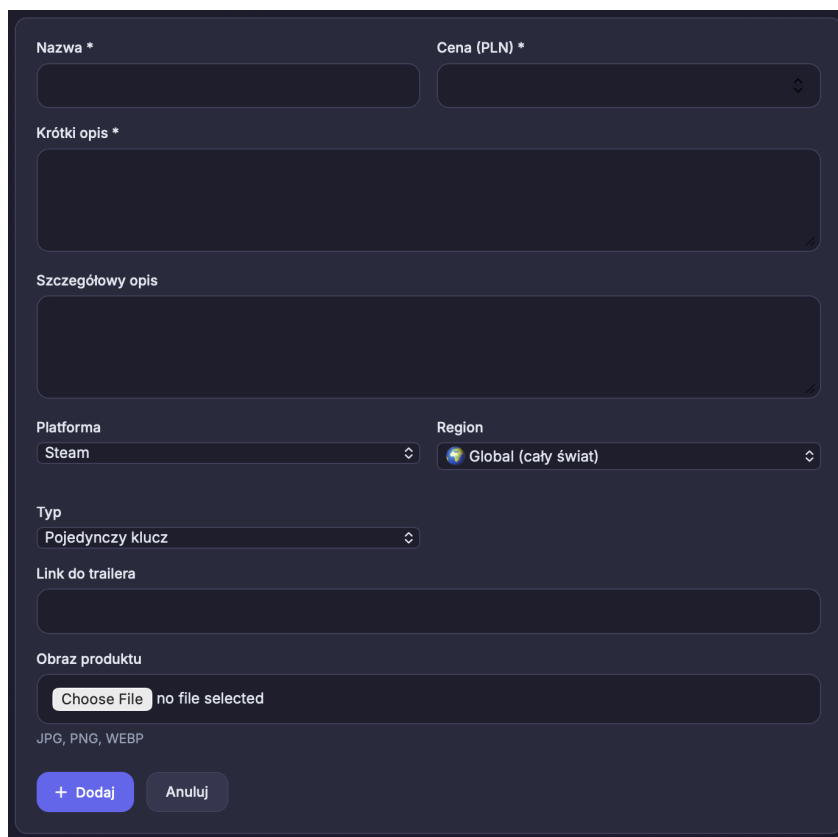
    rows.forEach(row => {
      const text = row.textContent.toLowerCase();
      const match = searchTerm === '' || text.includes(searchTerm);
      row.style.display = match ? '' : 'none';
      if (match) visibleCount++;
    });

    // Komunikat o braku wyników
    let noResults = table.querySelector('.no-search-results');
    if (visibleCount === 0 && searchTerm !== '') {
      if (!noResults) {
        noResults = document.createElement('tr');
        noResults.className = 'no-search-results';
        var td = document.createElement('td');
        td.setAttribute('colspan', '10');
        var icon = document.createElement('i');
        icon.className = 'fas fa-search';
        td.appendChild(icon);
        td.appendChild(document.createTextNode('Brak wyników dla "' + searchTerm
          + '"'));
        noResults.appendChild(td);
        table.querySelector('tbody').appendChild(noResults);
      }
    } else if (noResults) {
      noResults.remove();
    }
  });

  // Skrót klawiszowy Ctrl+K / Cmd+K
  document.addEventListener('keydown', function(e) {
    if ((e.ctrlKey || e.metaKey) && e.key === 'k') {
      e.preventDefault();
      input.focus();
      input.select();
    }
  });
}
```

```
// Inicjalizacja po załadowaniu strony
document.addEventListener('DOMContentLoaded', function() {
    quickSearch('searchProducts', 'productsTable');
});
```

Poza przeglądaniem i filtrowaniem listy, administrator ma również możliwość dodawania nowych produktów do bazy danych. Po kliknięciu przycisku „Dodaj produkt” w panelu administracyjnym w adresie URI ustawiany jest parametr `action=new_product`, a PHP na tej podstawie wyświetla pusty formularz HTML (rys. 3.12). W formularzu ustawiony jest atrybut `enctype="multipart/form-data"` wymagany do przesyłania plików. Bez niego przeglądarka nie wysyła danych binarnych (np. obrazu okładki). W formularzu znajduje się również pole `csrfField()`, które wstawia ukryty token CSRF. Jest to zabezpieczenie przed atakiem CSRF (Cross-Site Request Forgery), czyli sytuacją, w której zewnętrzna strona mogłaby wymusić wykonanie żądania w imieniu zalogowanego administratora. Następnie administrator uzupełnia dane produktu: nazwę, cenę, krótki i szczegółowy opis, platformę, region, typ produktu, link do trailera oraz dodaje plik graficzny (jako banner produktu). Po zatwierdzeniu formularza dane są wysyłane metodą POST na stronę `admin.php` z parametrem `action=add_product`, gdzie następuje ich dalsze przetwarzanie po stronie serwera.



The image shows a dark-themed web form for adding a new product. The form contains the following fields and controls:

- Nazwa ***: A text input field.
- Cena (PLN) ***: A text input field with a dropdown arrow.
- Krótki opis ***: A large text area.
- Szczegółowy opis**: A large text area.
- Platforma**: A dropdown menu with "Steam" selected.
- Region**: A dropdown menu with "Global (cały świat)" selected.
- Typ**: A dropdown menu with "Pojedynczy klucz" selected.
- Link do trailera**: A text input field.
- Obraz produktu**: A file upload section with a "Choose File" button and the text "no file selected". Below it, the supported formats "JPG, PNG, WEBP" are listed.
- Buttons**: At the bottom, there are two buttons: "+ Dodaj" (highlighted in blue) and "Anuluj".

Rysunek 3.12: Pusty formularz w panelu administratora

```
<!-- HTML - formularz nowego produktu -->
<form action="admin.php?action=add_product" method="post" enctype="multipart/form-data">
  <?php echo csrfField(); ?> <!-- token CSRF jako ukryte pole -->

  <div class="form-grid">
    <div class="form-group">
      <label class="form-label">Nazwa *</label>
      <input type="text" name="name" class="form-input" required>
    </div>
    <div class="form-group">
      <label class="form-label">Cena (PLN) *</label>
      <input type="number" step="0.01" name="price" class="form-input" required>
    </div>
  </div>

  <div class="form-group">
    <label class="form-label">Krótki opis *</label>
    <textarea name="description" class="form-textarea" rows="2" required></textarea>
  </div>

  <div class="form-group">
    <label class="form-label">Platforma</label>
    <select name="platform" class="form-select">
      <option value="Steam">Steam</option>
      <option value="Epic Games">Epic Games</option>
      <option value="Xbox">Xbox</option>
      <option value="PlayStation">PlayStation</option>
      <!-- ... inne platformy -->
    </select>
  </div>

  <div class="form-group">
    <label class="form-label">Typ</label>
    <select name="type" class="form-select">
      <option value="single">Pojedynczy klucz</option>
      <option value="bundle">Pakiet</option>
    </select>
  </div>

  <div class="form-group">
    <label class="form-label">Obraz produktu</label>
    <input type="file" name="product_image" class="form-input" accept="image/*">
  </div>

  <button type="submit" class="btn btn-primary">Dodaj</button>
</form>
```

Po wysłaniu formularza metodą POST (akcja `action=add_product`) PHP najpierw weryfikuje token CSRF za pomocą funkcji `requireCsrf()`. Następnie system odczytuje dane z tablicy `$_POST`, a przesłany plik graficzny jest przetwarzany przez funkcję pomocniczą `processImageUpload()` znajdującą się w bibliotece `utils.php`. Funkcja ta sprawdza rozszerzenie pliku oraz jego rzeczywisty typ MIME. Sprawdzenie MIME po stronie serwera jest ważne, ponieważ chroni przed podszywaniem się pod obraz poprzez samą zmianę rozszerzenia pliku. Jeżeli walidacja zakończy się pomyślnie, funkcja zwraca binarne dane obrazka. Następnie PHP przygotowuje zapytanie SQL w formie *prepared statement*, czyli zapytania sparametryzowanego. Jest to zabezpieczenie przed atakiem SQL Injection, ponieważ dane użytkownika nie są bezpośrednio wklejane do treści zapytania SQL. Binarne dane obrazka są przesyła-

ne oddzielnie metodą `send_long_data()`, ponieważ pole typu BLOB (Binary Large Object) może mieć bardzo duży rozmiar. Po poprawnym wykonaniu `execute()` administrator zostaje przekierowany z powrotem na listę produktów z komunikatem `msg=added`.

```
// PHP - walidacja obrazka w utils.php
function processImageUpload($fieldName = 'product_image') {
    if (!isset($_FILES[$fieldName]) || $_FILES[$fieldName]['error'] !== UPLOAD_ERR_OK) {
        return null;
    }

    $imageFileType = strtolower(pathinfo($_FILES[$fieldName]['name'],
        PATHINFO_EXTENSION));
    if (!in_array($imageFileType, ALLOWED_IMAGE_EXTENSIONS)) {
        return null;
    }

    // Walidacja rzeczywistego MIME
    $finfo = finfo_open(FILEINFO_MIME_TYPE);
    $mimeType = finfo_file($finfo, $_FILES[$fieldName]['tmp_name']);
    finfo_close($finfo);

    $allowedMimes = ['image/jpeg', 'image/png', 'image/gif', 'image/webp'];
    if (!in_array($mimeType, $allowedMimes)) {
        return null;
    }

    return [
        'data' => file_get_contents($_FILES[$fieldName]['tmp_name']),
        'mime_type' => $mimeType
    ];
}

// PHP - zapis nowego produktu do bazy danych
if ($action === 'add_product' && $_SERVER['REQUEST_METHOD'] === 'POST') {
    requireCsrf('admin.php?action=products'); // weryfikacja tokenu CSRF

    $name = $_POST['name'] ?? '';
    $price = $_POST['price'] ?? 0;
    $description = $_POST['description'] ?? '';
    $platform = $_POST['platform'] ?? 'Steam';
    $region = $_POST['region'] ?? 'Global';
    $type = $_POST['type'] ?? 'single';

    $image_data = null;
    $mime_type = null;
    $uploadedImage = processImageUpload('product_image');
    if ($uploadedImage) {
        $image_data = $uploadedImage['data'];
        $mime_type = $uploadedImage['mime_type'];
    }

    // Prepared statement - ochrona przed SQL Injection
    $sql = "INSERT INTO products
        (name, price, description, detailed_description, trailer_url,
         image_data, mime_type, platform, region, type)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    $stmt = $conn->prepare($sql);
    $null = null; // placeholder dla BLOB - dane wysyłane przez send_long_data()
    $stmt->bind_param("sdsssbssss",
        $name, $price, $description, $detailed_description,
        $trailer_url, $null, $mime_type, $platform, $region, $type
    );
}
```

```

if ($image_data !== null) {
    $stmt->send_long_data(5, $image_data);
}

if ($stmt->execute()) {
    header("Location: admin.php?action=products&msg=added");
    exit();
}
}

```

Gdy administrator klika ikonę edycji przy wybranym produkcie, przeglądarka otwiera stronę `admin.php` z parametrami `action=edit_product` oraz odpowiednio ustawionym `id=X`. Na początku obsługi żądania (jeszcze przed wygenerowaniem kodu HTML) PHP pobiera z bazy danych rekord o podanym identyfikatorze i zapisuje go do zmiennej `$editProduct`. Następnie pobrane wartości są wstawiane do pól formularza jako wartości domyślne (`value=`), dzięki czemu administrator widzi aktualne dane i może je od razu zmodyfikować. W przypadku list rozwijanych PHP porównuje każdą opcję z aktualną wartością pola i dodaje atrybut `selected`, aby zaznaczyć właściwą opcję. Jeżeli produkt posiada zapisany obraz, system wyświetla jego podgląd oraz udostępnia opcję usunięcia poprzez pole wyboru `remove_image`.

```

// PHP - wczytanie danych produktu przed renderowaniem formularza
$editProduct = null;
if ($action === 'edit_product' && isset($_GET['id'])) {
    $id = (int)$_GET['id']; // rzutowanie na int - zapobiega SQL Injection
    $stmt = $conn->prepare("SELECT * FROM products WHERE id = ?");
    $stmt->bind_param("i", $id);
    $stmt->execute();
    $result = $stmt->get_result();
    if ($result) $editProduct = $result->fetch_assoc();
    $stmt->close();
}

<!-- HTML - formularz edycji z wczytanymi danymi -->
<form action="admin.php?action=update_product&id=<?php echo $editProduct['id']; ?>"
      method="post" enctype="multipart/form-data">
    <?php echo csrfField(); ?>

    <input type="text" name="name" class="form-input"
      value="<?php echo htmlspecialchars($editProduct['name']); ?>" required>

    <input type="number" step="0.01" name="price" class="form-input"
      value="<?php echo $editProduct['price']; ?>" required>

    <select name="platform" class="form-select">
      <?php foreach (['Steam', 'Epic Games', 'Xbox', 'PlayStation'] as $plat): ?>
        <option value="<?php echo $plat; ?>"
          <?php echo ($editProduct['platform'] ?? '') === $plat ? 'selected' : ''; ?>>
          <?php echo $plat; ?>
        </option>
      <?php endforeach; ?>
    </select>

    <!-- Podgląd obecnego obrazka z opcją usunięcia -->
    <?php if (!empty($editProduct['image_data'])): ?>
    <div class="image-preview">
      
      <div class="checkbox-group">

```

```

        <input type="checkbox" name="remove_image" value="1" id="rm">
        <label for="rm">Usuń obraz</label>
    </div>
</div>
<?php endif; ?>
<input type="file" name="product_image" class="form-input" accept="image/*">
<button type="submit" class="btn btn-primary">Zapisz</button>
</form>

```

Po wysłaniu formularza edycji produktu za pomocą metody POST (akcja `action=update_product`) PHP ponownie weryfikuje token CSRF. Następnie system pobiera z bazy danych aktualny obraz produktu, ponieważ jest to potrzebne do poprawnej obsługi sytuacji, gdy administrator nie wybierze nowego pliku. W tym miejscu zastosowano logikę trójstanową. Po pierwsze, jeżeli został przesłany nowy plik, system zapisuje nowy obraz. Po drugie, jeżeli zaznaczono pole wyboru `remove_image`, obraz produktu jest usuwany (ustawiany jako NULL). W pozostałych przypadkach dotychczasowy obraz pozostaje bez zmian. Na końcu PHP wykonuje zapytanie UPDATE w formie *prepared statement*, aktualizując dane produktu w bazie danych, a następnie przekierowuje administratora na listę produktów z komunikatem `msg=updated`.

```

// PHP - aktualizacja produktu
if ($action === 'update_product'
    && $_SERVER['REQUEST_METHOD'] === 'POST' && isset($_GET['id'])) {
    requireCsrf('admin.php?action=products');
    $id = (int)$_GET['id'];

    // ... odczyt $_POST analogiczny jak przy dodawaniu ...

    // Krok 1: pobierz obecny obraz (fallback, gdy nie przesłano nowego)
    $stmt = $conn->prepare("SELECT image_data, mime_type FROM products WHERE id = ?");
    $stmt->bind_param("i", $id);
    $stmt->execute();
    $currentProduct = $stmt->get_result()->fetch_assoc();
    $stmt->close();

    $image_data = $currentProduct['image_data'] ?? null;
    $mime_type = $currentProduct['mime_type'] ?? null;

    // Krok 2: nowy plik - zastąp
    $uploadedImage = processImageUpload('product_image');
    if ($uploadedImage) {
        $image_data = $uploadedImage['data'];
        $mime_type = $uploadedImage['mime_type'];
    }

    // Krok 3: checkbox "Usuń" - wyzeruj
    if (isset($_POST['remove_image'])) {
        $image_data = null;
        $mime_type = null;
    }

    // Zapis do bazy
    $sql = "UPDATE products
        SET name=?, price=?, description=?, detailed_description=?,
            trailer_url=?, image_data=?, mime_type=?,
            platform=?, region=?, type=?
        WHERE id=?";
    $stmt = $conn->prepare($sql);
    $null = null;

```

```

$stmt->bind_param("sdssbssssi",
    $name, $price, $description, $detailed_description,
    $trailer_url, $null, $mime_type, $platform, $region, $type, $id
);

if ($image_data !== null) {
    $stmt->send_long_data(5, $image_data);
}

if ($stmt->execute()) {
    header("Location: admin.php?action=products&msg=updated");
    exit();
}
}

```

3.5 Zarządzanie kluczami w panelu administratora

Po przejściu do zakładki „Klucze” (akcja `action=keys`) serwer jako pierwsze wykonuje zapytanie agregujące, które grupuje wszystkie klucze z tabeli `product_keys` według gry oraz regionu (rys. 3.13).

GRA	REGION	DOSTĘPNE	SPRZEDANE	ŁĄCZNIE
ARC Raiders (LT)	LITWA	0	14	14
ARC Raiders (PL)	POLSKA	0	5	5
ARC Raiders (PL)	POLSKA	9	1	10
Battlefield 6	NIEMCY	9	1	10
Cyberpunk 2077	GLOBAL	0	1	1

Rysunek 3.13: Stan magazynowy kluczy w panelu administratora

Funkcja `COUNT(*)` w SQL zlicza łączną liczbę kluczy, natomiast wyrażenie `SUM(CASE WHEN is_sold = 0 ...)` oblicza, ile z nich jest jeszcze dostępnych (niesprzedanych). Otrzymany wynik jest wyświetlany w tabeli „Stan magazynowy”, gdzie każdy wiersz pokazuje nazwę gry, region oraz trzy liczniki: dostępne (zielona odznaka), sprzedane (szara odznaka) oraz łącznie. Dzięki temu administrator może jednym rzutem oka sprawdzić, dla których gier zaczyna brakować kluczy w magazynie.

```

// PHP - zapytanie agregujące stany magazynowe kluczy
$statQuery = "SELECT p.name, pk.region,
    COUNT(*) as total,
    SUM(CASE WHEN pk.is_sold = 0 THEN 1 ELSE 0 END) as available
FROM product_keys pk
LEFT JOIN products p ON pk.product_id = p.id
GROUP BY pk.product_id, pk.region
ORDER BY p.name, pk.region";

```

```

$statResult = $conn->query($statQuery);

if ($statResult) while ($s = $statResult->fetch_assoc()) $keysStats[] = $s;

<!-- HTML - tabela stanu magazynowego -->
<table class="table">
  <thead>
    <tr><th>Gra</th><th>Region</th><th>Dostępne</th>
      <th>Sprzedane</th><th>Łącznie</th></tr>
    </thead>
    <tbody>
      <?php foreach ($keysStats as $stat): ?>
        <tr>
          <td><strong><?php echo htmlspecialchars($stat['name'] ?? 'Nieznana');
            ?></strong></td>
          <td><span class="badge badge-info"><?php echo
            htmlspecialchars($stat['region']); ?></span></td>
          <td><span class="badge badge-success"><?php echo $stat['available'];
            ?></span></td>
          <td><span class="badge badge-secondary"><?php echo $stat['total'] -
            $stat['available']; ?></span></td>
          <td><?php echo $stat['total']; ?></td>
        </tr>
      <?php endforeach; ?>
    </tbody>
  </table>

```

Trochę poniżej stanu magazynowanego kluczy znajduje się formularz dodawania kluczy który umożliwi wklejenie wielu kluczy jednocześnie. Każdy klucz znajduje się w osobnej linii w polu tekstowym typu `textarea` (rys. 3.14).

Rysunek 3.14: Dodanie nowego klucza poprzez panel administratora

Administrator wybiera grę z listy rozwijanej, która jest zasilana wynikiem zapytania `SQL SELECT id, name FROM products`, a następnie wskazuje region dla dodawanych kluczy. Po stronie serwera PHP pobiera treść pola z kluczami i rozdziela ją na linie za pomocą funkcji `explode("\n", $keys_text)`. Następnie funkcje `array_filter()` oraz `array_map('trim', ...)` usuwają puste linie i zbędne białe znaki. Dla każdego niepustego klucza wykonywane jest osobne zapytanie `INSERT` z użyciem *prepared statement*. Zmienna `$added` zlicza, ile kluczy zostało faktycznie zapisanych w bazie danych, a jej wartość jest przekazywana w komunikacie po przekierowaniu administratora.

```
// PHP - przetwarzanie wsadowego dodawania kluczy
if ($action === 'add_keys' && $_SERVER['REQUEST_METHOD'] === 'POST') {
    requireCsrf('admin.php?action=keys');
    $product_id = (int)$_POST['product_id'];
    $region = $_POST['region'] ?? 'Global';
    $keys_text = $_POST['keys'] ?? '';

    // Rozbij tekst na tablicę, usuń puste linie i białe znaki
    $keys = array_filter(array_map('trim', explode("\n", $keys_text)));
    $added = 0;

    foreach ($keys as $key) {
        if (!empty($key)) {
            $stmt = $conn->prepare(
                "INSERT INTO product_keys (product_id, game_key, region) VALUES (?, ?, ?)"
            );
            $stmt->bind_param("iss", $product_id, $key, $region);
            if ($stmt->execute()) $added++;
            $stmt->close();
        }
    }

    header("Location: admin.php?action=keys&msg=added&count=$added");
    exit();
}

<!-- HTML - formularz dodawania kluczy -->
<form action="admin.php?action=add_keys" method="post" id="addKeysForm">
    <?php echo csrfField(); ?>
    <div class="form-grid">
        <div class="form-group">
            <label class="form-label">Gra *</label>
            <select name="product_id" class="form-select" id="keyProductSelect" required>
                <option value="">-- Wybierz grę --</option>
                <?php foreach ($allProducts as $p): ?>
                    <option value="<?php echo $p['id']; ?>">
                        <?php echo htmlspecialchars($p['name']); ?> - <?php echo
                            number_format($p['price'], 2); ?> zł
                    </option>
                <?php endforeach; ?>
            </select>
        </div>
        <div class="form-group">
            <label class="form-label">Region kluczy *</label>
            <select name="region" class="form-select" id="keyRegionSelect" required>
                <?php echo getRegionOptions(); ?>
            </select>
        </div>
    </div>
    <div class="form-group">
        <label class="form-label">Klucze (jeden klucz per linia) *</label>
        <textarea name="keys" class="form-textarea" rows="6"
            placeholder="XXXXX-XXXXX-XXXXX&#10;YYYYY-YYYYY-YYYYY"
            required></textarea>
    </div>
    <button type="submit" class="btn btn-primary" id="addKeysBtn">Dodaj klucze</button>
</form>
```

Pod formularzem dodawania kluczy znajduje się tabela zawierająca wszystkie zapisane klucze (rys. 3.15).

ID	GRA	KLUCZ	REGION	STATUS	DODANO	AKCJE
#184	Resident Evil 4 (DLC-Separate Ways) 4712 zl	*****-*****-8ML4R	POLSKA	DOSTĘPNY	27.01.2026 15:27	[ikona]
#169	Resident Evil 4 (DLC-Separate Ways) 4712 zl	*****-*****-8XWSP	POLSKA	DOSTĘPNY	27.01.2026 15:27	[ikona]
#170	Resident Evil 4 (DLC-Separate Ways) 4712 zl	*****-*****-LX8AF	POLSKA	DOSTĘPNY	27.01.2026 15:27	[ikona]

Rysunek 3.15: Lista dodanych kluczy poprzez panel administratora

W przeciwieństwie do wyszukiwarki produktów, która działa po stronie przeglądarki, filtrowanie kluczy odbywa się po stronie serwera. PHP dynamicznie buduje zapytanie SQL, dodając warunki `WHERE` wyłącznie dla tych filtrów, które zostały faktycznie ustawione (gra, region, status). Takie rozwiązanie zapewnia poprawność wyników nawet przy dużej liczbie rekordów w bazie danych. Każdy wiersz tabeli zawiera identyfikator klucza, nazwę gry, zamaskowaną wartość klucza, region, status (dostępny lub sprzedany) oraz datę jego dodania. Dzięki temu administrator może w przejrzysty sposób zarządzać stanem magazynowym i kontrolować historię dodawanych kluczy.

```
// PHP - dynamiczne budowanie zapytania z filtrami
$filterProduct = isset($_GET['product_id']) ? (int)$_GET['product_id'] : 0;
$filterRegion = isset($_GET['region']) ? $_GET['region'] : '';
$filterStatus = isset($_GET['status']) ? $_GET['status'] : '';

$keysQuery = "SELECT pk.*, p.name as product_name, p.price as product_price
FROM product_keys pk
LEFT JOIN products p ON pk.product_id = p.id WHERE 1=1";
$params = [];
$types = "";

// Dodawaj kolejne warunki tylko gdy filtr jest ustawiony
if ($filterProduct > 0) {
    $keysQuery .= " AND pk.product_id = ?";
    $params[] = $filterProduct; $types .= "i";
}
if (!empty($filterRegion)) {
    $keysQuery .= " AND pk.region = ?";
    $params[] = $filterRegion; $types .= "s";
}
if ($filterStatus === 'available') $keysQuery .= " AND pk.is_sold = 0";
if ($filterStatus === 'sold') $keysQuery .= " AND pk.is_sold = 1";
$keysQuery .= " ORDER BY pk.created_at DESC";

$stmt = $conn->prepare($keysQuery);
if (!empty($params)) $stmt->bind_param($types, ...$params);
$stmt->execute();
$keysResult = $stmt->get_result();
while ($k = $keysResult->fetch_assoc()) $keys[] = $k;

<!-- HTML - formularz filtrowania (wysła GET, odświeża stronę) -->
<form method="get">
    <input type="hidden" name="action" value="keys">
    <select name="product_id" class="form-select">
        <option value="">Wszystkie gry</option>
```

```

<?php foreach ($allProducts as $p): ?>
<option value="<?php echo $p['id']; ?>"
    <?php echo $filterProduct == $p['id'] ? 'selected' : ''; ?>
    <?php echo htmlspecialchars($p['name']); ?>
</option>
<?php endforeach; ?>
</select>
<select name="status" class="form-select">
    <option value="">Wszystkie statusy</option>
    <option value="available" <?php echo $filterStatus === 'available' ? 'selected' : ''; ?>>Dostępne</option>
    <option value="sold" <?php echo $filterStatus === 'sold' ? 'selected' : ''; ?>>Sprzedane</option>
</select>
<button type="submit" class="btn btn-secondary">Filtruj</button>
</form>

```

3.6 Zarządzanie systemem rabatowym w panelu administratora

Zarządzanie systemem rabatowym w panelu administratora odbywa się w zakładce „Kody rabatowe” (akcja `action=promo_codes`). Tabela `promo_codes` przechowuje każdy kod jako unikalny ciąg znaków wraz z wartością procentową zniżki (`discount_percent`), flagą aktywności (`is_active`) oraz datą utworzenia. Po wejściu do zakładki serwer pobiera wszystkie kody zapytaniem `SELECT * FROM promo_codes ORDER BY created_at DESC` i wyświetla je w tabeli, gdzie każdy wiersz pokazuje identyfikator, kod, wysokość rabatu, aktualny status (zielona odznaka „Aktywny” lub szara „Nieaktywny”) oraz datę dodania (rys. 3.16).

ID	KOD	RABAT	STATUS	DATA UTWORZENIA	AKCJE
#4	TEST2	49%	AKTYWNY	11.02.2026 21:15	 
#3	TEST	10%	NIEAKTYWNY	08.02.2026 21:23	 

Rysunek 3.16: Tabela z kodami promocyjnymi w panelu administratora

Nad tabelą umieszczony jest formularz (rys. 3.17) pozwalający dodać nowy kod: administrator wpisuje jego nazwę oraz wartość rabatu w przedziale 1-100%. Po zatwierdzeniu serwer wykonuje akcję `add_promo_code`, to znaczy, po weryfikacji tokena CSRF kod jest normalizowany za pomocą funkcji `strtoupper(trim(...))` i wstawiany poleceniem `INSERT`. Unikalność kodu wymuszana jest ograniczeniem `UNIQUE` na poziomie bazy danych. Próba dodania duplikatu kończy się przekierowaniem z parametrem `msg=duplicate`. Każdy istniejący kod promocyjny można dezaktywować lub ponownie aktywować jednym kliknięciem. Służy do tego akcja o `toggle_promo_code`. Wykonuje ona atomową operację `SET is_active = NOT is_active`, co pozwala

tymczasowo wyłączyć promocję bez jej usuwania. Usunięcie kodu promocyjnego jest możliwe poprzez akcję `delete_promo_code` z potwierdzeniem po stronie klienta (`onclick="return confirm(...)"`).

Rysunek 3.17: Pole dodania nowego kodu rabatowego w panelu administratora

Po stronie sklepu kod promocyjny walidowany jest za pomocą funkcji `validatePromoCode()`. Funkcja ta wykonuje zapytanie SQL sprawdzające, czy kod promocyjny istnieje i jest aktywny, a następnie zwraca tablicę o trzech elementach: `['valid', 'discount', 'id']`. Zastosowany rabat zapisywany jest w sesji użytkownika (klucze `applied_promo_code`, `applied_discount`, `applied_promo_code_id`), dzięki czemu przetrwa między odświeżeniami strony koszyka. Podczas finalizowania zamówienia serwer odczytuje rabat z sesji i oblicza końcową kwotę do zapłaty:

$$\text{total}_{\text{końcowe}} = \text{total} - \text{total} \times \frac{\text{discount}}{100}$$

gdzie `discount` jest wartością procentową przyznanego rabatu. Wynikowa kwota oraz użyty kod są zapisywane bezpośrednio w rekordzie tabeli `orders` (kolumny `promo_code`, `discount_percent`, `promo_code_id`). Po pomyślnym zakupie dane kodu są czyszczone z sesji.

```
// PHP - walidacja kodu promocyjnego
function validatePromoCode($conn, $code) {
    $stmt = $conn->prepare(
        "SELECT id, discount_percent
        FROM promo_codes
        WHERE code = ? AND is_active = 1"
    );
    $stmt->bind_param("s", $code);
    $stmt->execute();
    $stmt->bind_result($id, $discount);
    if ($stmt->fetch())
        return ['valid' => true, 'discount' => (int)$discount, 'id' => (int)$id];
    return ['valid' => false, 'discount' => 0, 'id' => null];
}

// PHP - zastosowanie rabatu podczas checkout
if ($promoResult['valid']) {
    $discount = $promoResult['discount'];
    $appliedCode = $formPromoCode;
    $promoCodeId = $promoResult['id'];
}
if ($discount > 0)
    $totalSum = $totalSum - ($totalSum * $discount / 100);

$orderSql = "INSERT INTO orders
            (user_id, total, promo_code, discount_percent, promo_code_id)
```

```

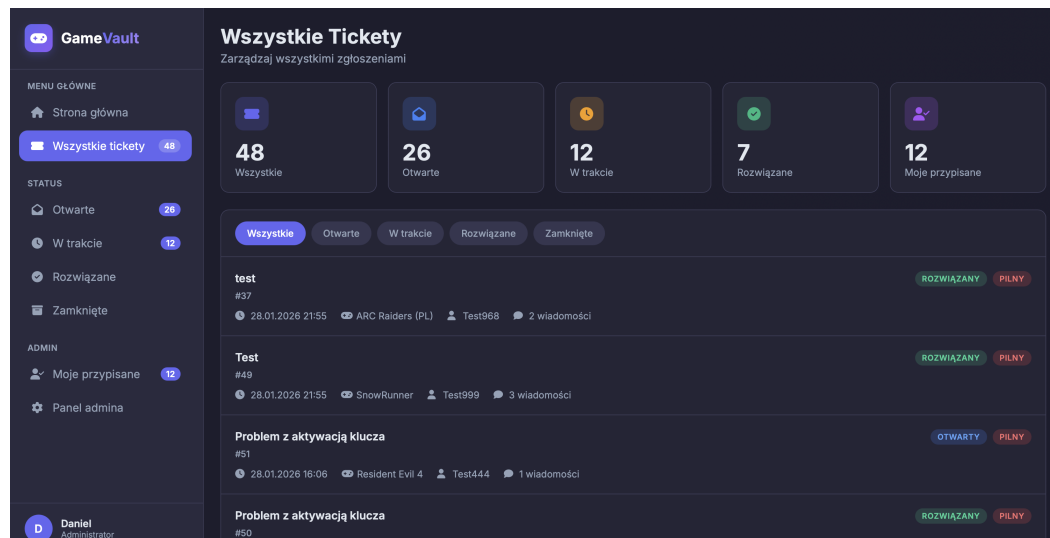
VALUES (?, ?, ?, ?, ?)";

<!-- HTML - wiersz tabeli kodów rabatowych -->
<tr>
  <td><strong>#<?php echo $pc['id']; ?></strong></td>
  <td><code><?php echo htmlspecialchars($pc['code']); ?></code></td>
  <td><strong><?php echo $pc['discount_percent']; ?>%</strong></td>
  <td>
    <?php if ($pc['is_active']): ?>
      <span class="badge badge-success">Aktywny</span>
    <?php else: ?>
      <span class="badge badge-secondary">Nieaktywny</span>
    <?php endif; ?>
  </td>
  <td><?php echo date('d.m.Y H:i', strtotime($pc['created_at'])); ?></td>
  <td>
    <a href="admin.php?action=toggle_promo_code&id=<?php echo $pc['id']; ?>"
      class="btn btn-sm">
      <i class="fas fa-toggle-<?php echo $pc['is_active'] ? 'off' : 'on'; ?>"></i>
    </a>
    <a href="admin.php?action=delete_promo_code&id=<?php echo $pc['id']; ?>"
      class="btn btn-danger btn-sm"
      onclick="return confirm('Usunac kod?');">
      <i class="fas fa-trash"></i>
    </a>
  </td>
</tr>

```

3.7 Zarządzanie systemem zgłoszeń w panelu administratora

W panelu administracyjnym istnieje w pełni funkcjonalny system zgłoszeń, do którego administrator przechodzi przez zakładkę „Tickety” w pasku bocznym. Widoczna jest tam dynamiczna odznaka z liczbą aktualnie otwartych zgłoszeń. Administrator widzi **wszystkie** zgłoszenia w systemie – użytkownik tylko własne. Lista jest sortowana priorytetowo (pilne → wysokie → średnie → niskie), a następnie według daty aktualizacji. Dostępne są filtry zakładkowe według statusu oraz widok „Moje przypisane”, ograniczający wyniki do zgłoszeń z `assigned_admin_id` ustawionym na bieżącego administratora. Statystyki na górze strony pokazują łączną liczbę zgłoszeń oraz rozbiecie na stany, a panel administratora ma dodatkowy kafelek zliczający aktywne zgłoszenia przypisane do niego (rys. 3.18).

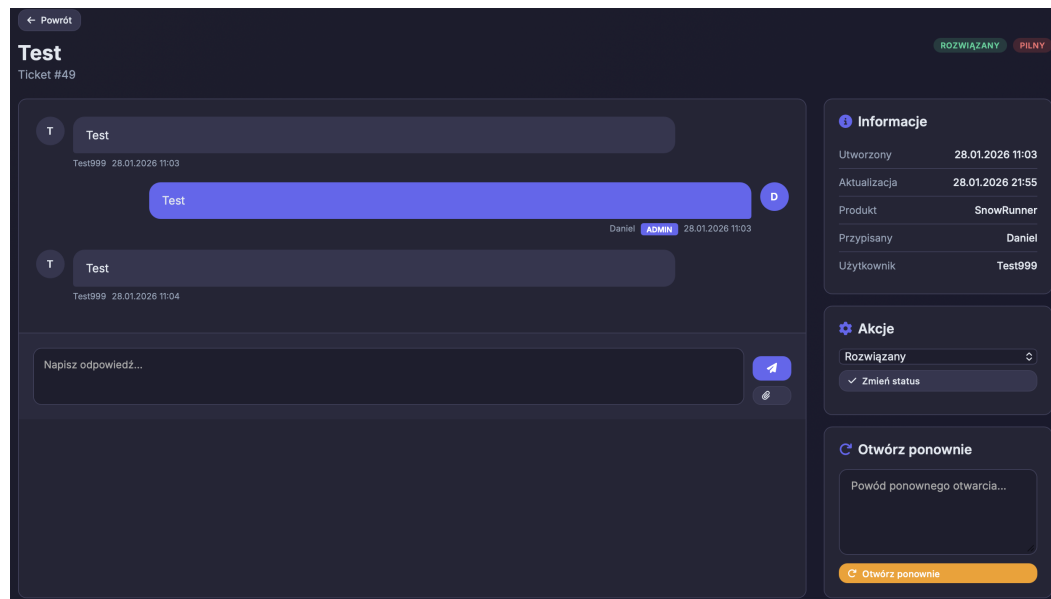


Rysunek 3.18: Główne okno systemu zgłoszeń od strony administratora

W widoku szczegółów administrator ma w prawym pasku bocznym panel akcji niedostępny dla użytkowników. Może on tam:

- zmienić status zgłoszenia na jeden z: `open`, `in_progress`, `resolved`, `closed`,
- przypisać zgłoszenie do siebie jednym kliknięciem, co automatycznie przełącza status na `in_progress`,
- ponownie otworzyć zamknięte zgłoszenie w operacji opakowanej w transakcję bazodanową, która zeruje przypisanie i wstawia do historii systemu komunikat z powodem.

Administrator może odpowiadać na każde zgłoszenie niezależnie od statusu. Każda jego odpowiedź jest oznaczona flagą `is_admin_reply = 1` i automatycznie przypisuje go jako opiekuna. Widok odświeża się co 15 sekund przez `fetch()` do endpointu `check_new_messages`, symulując komunikację w czasie rzeczywistym (rys. 3.19).



Rysunek 3.19: Okno chatu w systemie zgłoszeń od strony administratora

```
// Akcja assign - przypisanie admina do zgłoszenia
if ($action === 'assign' && $isAdmin && $_SERVER['REQUEST_METHOD'] === 'POST') {
    requireCsrf();
    $ticketId = (int)$_POST['ticket_id'];
    $sql = "UPDATE tickets SET assigned_admin_id = ?, status = 'in_progress',
        updated_at = CURRENT_TIMESTAMP WHERE id = ?";
    $stmt->bind_param("ii", $_SESSION['user_id'], $ticketId);
    $stmt->execute();
    header("Location: tickets.php?action=view&id=$ticketId");
}

// Akcja reply - efekt uboczny: auto-przypisanie admina
if ($isAdmin) {
    $updateSql = "UPDATE tickets SET status = 'in_progress',
        assigned_admin_id = ?, updated_at = CURRENT_TIMESTAMP WHERE id = ?";
    // ...
}

// Akcja reopen - transakcja: reset statusu + komunikat systemowy
$conn->begin_transaction();
$updateSql = "UPDATE tickets SET status = 'open', assigned_admin_id = NULL,
    updated_at = CURRENT_TIMESTAMP WHERE id = ?";
$reopenMessage = "Ticket został ponownie otwarty. Powód: " . $reopenReason;
$msgSql = "INSERT INTO ticket_messages (ticket_id, user_id, message, is_admin_reply)
    VALUES (?, ?, ?, ?)";
$conn->commit();

// JS - polling co 15 sekund
setInterval(function() {
    fetch('tickets.php?action=check_new_messages&ticket_id=<?php echo $ticketId; ?>')
        .then(r => r.json())
        .then(data => { if (data.hasNewMessages) location.reload(); });
}, 15000);
```

3.8 Integracja systemu płatności Stripe

Integracja z bramką płatniczą Stripe została zrealizowana przy użyciu wbudowanego w PHP rozszerzenia cURL, gdzie klucz tajny przekazywany jest jako dane uwierzytelniające HTTP Basic Auth.

Proces płatności przebiega w następujących etapach:

- po zatwierdzeniu koszyka system buduje listę pozycji `line_items` z cenami przeliczonymi na grosze oraz tworzy sesję Checkout w API Stripe (POST `/v1/checkout/sessions`), po czym przekierowuje użytkownika pod zwrócony adres URL,
- użytkownik finalizuje płatność kartą na hostowanej stronie Stripe,
- po pozytywnej autoryzacji Stripe przekierowuje użytkownika z powrotem do aplikacji z parametrem `session_id`, gdzie system weryfikuje status płatności (`payment_status == 'paid'`) poprzez zapytanie HTTP GET `/v1/checkout/sessions/{id}`; następnie w obrębie transakcji bazodanowej (`begin_transaction / commit`) zapisuje zamówienie wraz z identyfikatorami `stripe_session_id` oraz `stripe_payment_intent`, rezerwuje klucze gier z puli (blokada FOR UPDATE) oraz wysyła je do użytkownika pocztą elektroniczną przy użyciu biblioteki PHPMailer.

```
// Surowe żądanie cURL do Stripe API
curl_setopt($ch, CURLOPT_USERPWD, STRIPE_SECRET_KEY . ':' );
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
$response = json_decode(curl_exec($ch), true);

// Tworzenie sesji i przekierowanie do Stripe
$session = createStripeCheckoutSession($lineItems, $checkoutEmail, $userId, [...]);
header('Location: ' . $session['url']);

// Weryfikacja płatności i transakcja bazodanowa
$session = getStripeSession($sessionId); // GET /v1/checkout/sessions/{id}
if ($session['payment_status'] !== 'paid') { /* odrzuć */ }
$conn->begin_transaction();
// INSERT INTO orders ... stripe_session_id, stripe_payment_intent
// UPDATE product_keys SET is_sold = 1 ... FOR UPDATE
$conn->commit();
```

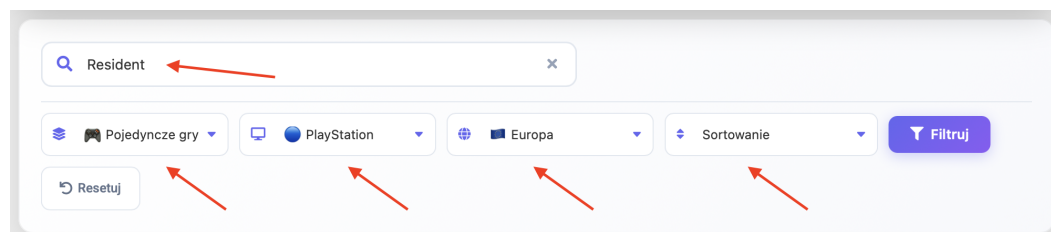
Rozdział 4

Testowanie i weryfikacja działania

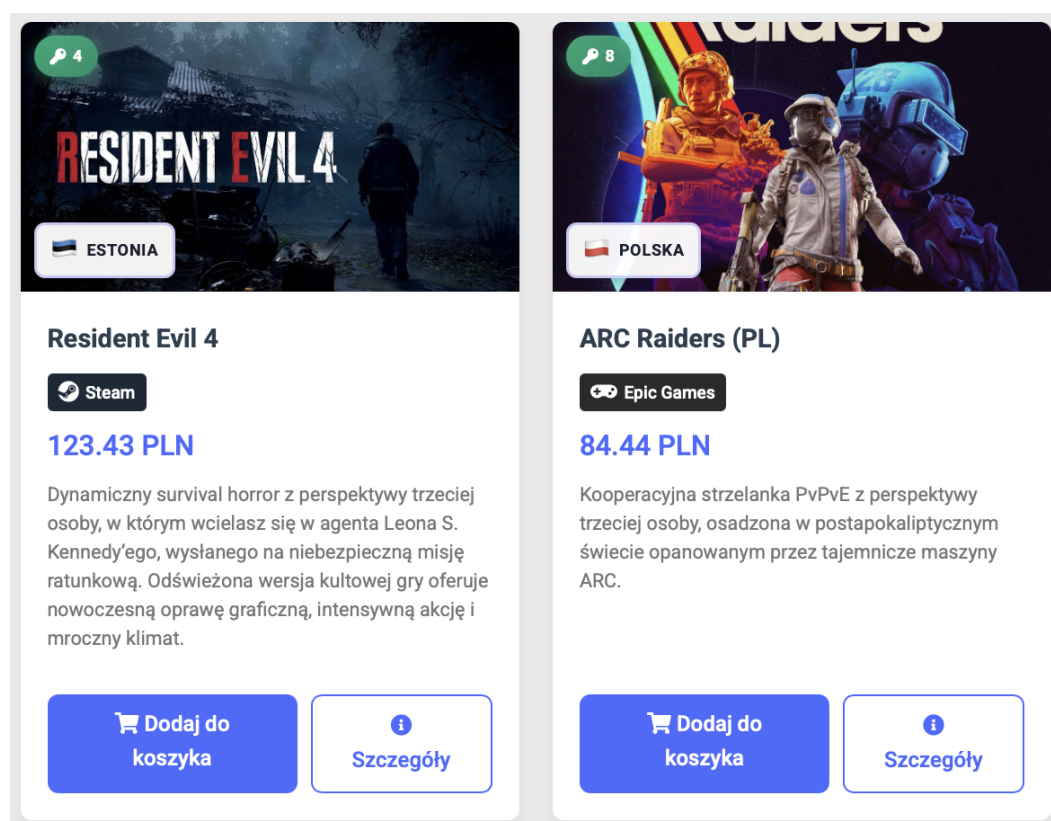
W tym rozdziale przeprowadzę główne testy oraz weryfikację, czy założone cele zostały poprawnie zrealizowane. Na podstawie wyników testów przedstawię również odpowiedzi na postawione pytania badawcze. W pierwszej kolejności zostanie sprawdzony standardowy przepływ użytkownika w aplikacji, czyli cały proces zakupu klucza do gry krok po kroku. Następnie zostanie zweryfikowane, czy równoległa prezentacja ofert skraca czas potrzebny na podjęcie decyzji oraz zmniejsza ryzyko pomyłki podczas wyboru. W dalszej części zostanie ocenione, które atrybuty karty oferty są dla użytkownika najbardziej informacyjne. Zostanie także sprawdzone, jak jawna prezentacja ceny końcowej (bez „dokręcania” dodatkowych opłat na etapie koszyka) wpływa na postrzeganą wiarygodność porównania. Na końcu przedstawiona zostanie tabela porównawcza pomiędzy moją platformą a oficjalnymi sklepami Steam oraz Epic Games, aby pokazać, na ile łatwiej i w niższej cenie można nabyć grę w moim systemie, w porównaniu do ręcznego porównywania ofert na różnych platformach.

4.1 Weryfikacja standardowego przepływu zakupu

W pierwszej kolejności klient rozpoczyna wyszukiwanie interesującego go produktu. Może skorzystać z pola wyszukiwania i wpisać nazwę potrzebnego tytułu lub użyć dedykowanych filtrów, aby dopasować wyniki do swoich konkretnych potrzeb (rys. 4.1). Następnie, dzięki wygodnemu porównaniu kart produktów, użytkownik może w czasie rzeczywistym przeglądać wybrane oferty obok siebie i świadomie wybrać najbardziej odpowiednią opcję (rys. 4.2).



Rysunek 4.1: Pole do wyszukiwania produktów z filtrami na stronie głównej



Rysunek 4.2: Karty gier obok siebie

Po wybraniu odpowiedniego produktu klient może od razu dodać go do koszyka lub przejść do strony ze szczegółami oferty. Na stronie produktu użytkownik ma możliwość obejrzenia traileru gry oraz zapoznania się z jej szczegółowym opisem, co pozwala mu podjąć bardziej świadomą decyzję zakupową (rys. 4.3, 4.4).

Kingdom Come: Deliverance II

 Epic Games

✓ Dostępnych: 2 klucze

Klucze dla regionów:

 Polska (2 klucze)

131.34 PLN

 Dodaj do koszyka

Realistyczna gra RPG osadzona w średniowiecznych Czechach, kontynuująca historię Henryka. Kingdom Come: Deliverance II oferuje rozbudowaną fabułę, otwarty świat oraz wymagający system walki oparty na realizmie.



Rysunek 4.3: Pierwsza część widoku szczegółów gry

Trailer gry



Szczegółowy opis

Platforma: Epic Games

Region aktywacji:  Polska

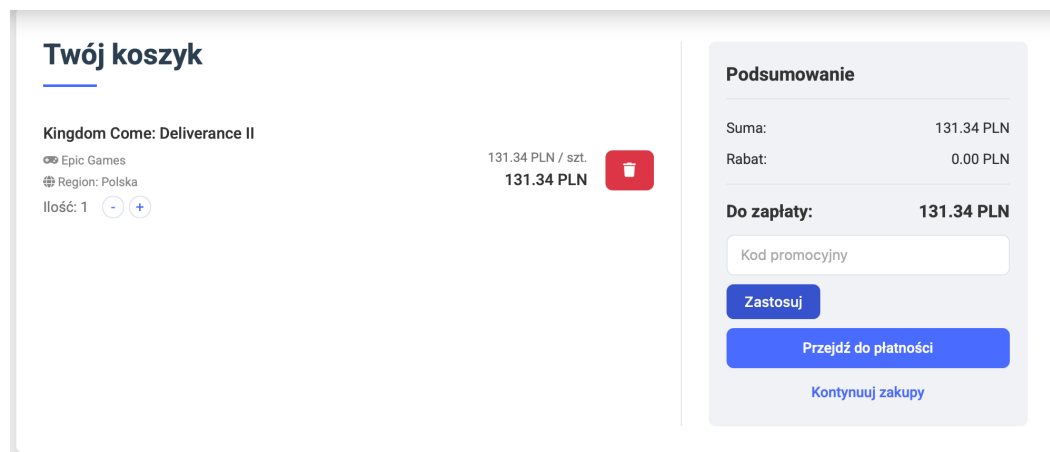
Kingdom Come: Deliverance II to bezpośrednia kontynuacja cenionego RPG, która ponownie przenosi graczy do realistycznie odwzorowanego świata średniowiecznej Europy Środkowej. Gracz wciela się w Henryka, syna kowala, którego losy ponownie zostają uwikłane w polityczne intryki, wojny i osobiste dramaty.

Gra kładzie duży nacisk na realizm historyczny – brak elementów fantasy, autentyczne uzbrojenie, wiarygodne lokacje oraz system rozwoju postaci oparty na zdobywanym doświadczeniu i podejmowanych decyzjach. Rozgrywka obejmuje eksplorację otwartego świata, wykonywanie zadań fabularnych i pobocznych, rozwijanie umiejętności oraz zaawansowany system walki wręcz i bronią białą.

Kingdom Come: Deliverance II oferuje rozbudowaną narrację, większą swobodę wyborów oraz jeszcze bardziej dopracowaną oprawę graficzną i mechaniki rozgrywki. To propozycja dla graczy szukających głębokiego, realistycznego RPG z naciskiem na historię i immersję.

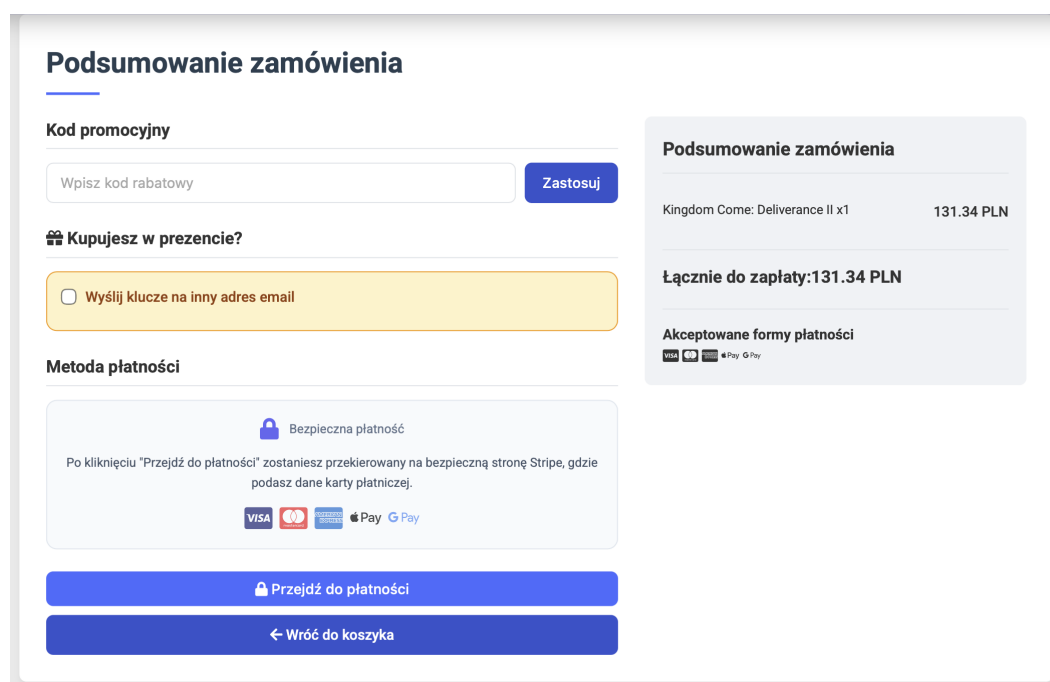
Rysunek 4.4: Druga część widoku szczegółów gry

Po dodaniu gry do koszyka klient zostaje automatycznie przekierowany do widoku koszyka, w którym widzi podstawowe informacje o wybranym produkcie, takie jak nazwa, platforma, region, ilość oraz cena. Na tym etapie użytkownik może również aktywować kod rabatowy oraz zapoznać się z całkowitą kwotą do zapłaty. Warto dodać, że kod promocyjny można wprowadzić także później, na etapie finalizacji zakupu (rys. 4.5).



Rysunek 4.5: Widok koszyka

Po kliknięciu przycisku „Przejdź do płatności” klient zostaje przekierowany do okna finalizacji zakupu. Na tym etapie ma jeszcze możliwość ponownego wpisania kodu promocyjnego, jeśli nie zrobił tego wcześniej w koszyku. Dodatkowo użytkownik może zmienić adres e-mail, na który zostanie wysłany klucz, co jest przydatne w przypadku zakupu prezentu dla innej osoby. W sekcji podsumowania klient widzi ponownie informacje o zakupionym produkcie oraz całkowitą kwotę do zapłaty. Ważnym elementem tego etapu jest również wyświetlenie akceptowanych form płatności. Obecnie mój system korzysta z integracji z partnerem Stripe poprzez API, który obsługuje płatności kartami Visa, MasterCard, American Express oraz metody Apple Pay i Google Pay (rys. 4.6).



Rysunek 4.6: Okno finalizacji zakupu

Po zatwierdzeniu płatności klient zostaje przekierowany za pośrednictwem API na stronę płatności operatora (rys. 4.7).

← TEST MODE

Kingdom Come: Deliverance II

PLN 131.34

Epic Games | Polska

Apple Pay Pay with link

OR

Contact information

Email game.vault.game.key@gmail.com

Payment method

Card

Card information

1234 1234 1234 1234 VISA

MM / YY CVC

Cardholder name

Full name on card

Country or region

Lithuania

Apple Pay

Save my information for faster checkout

Pay securely on this site and everywhere Link is accepted.

Pay

Powered by stripe Terms Privacy

Rysunek 4.7: Formularz opłaty po stronie partnera

Po pomyślnym zakończeniu płatności klient zostaje automatycznie przekierowany z powrotem na platformę. System wyświetla komunikat potwierdzający, że zakup został zrealizowany poprawnie (rys. 4.8).



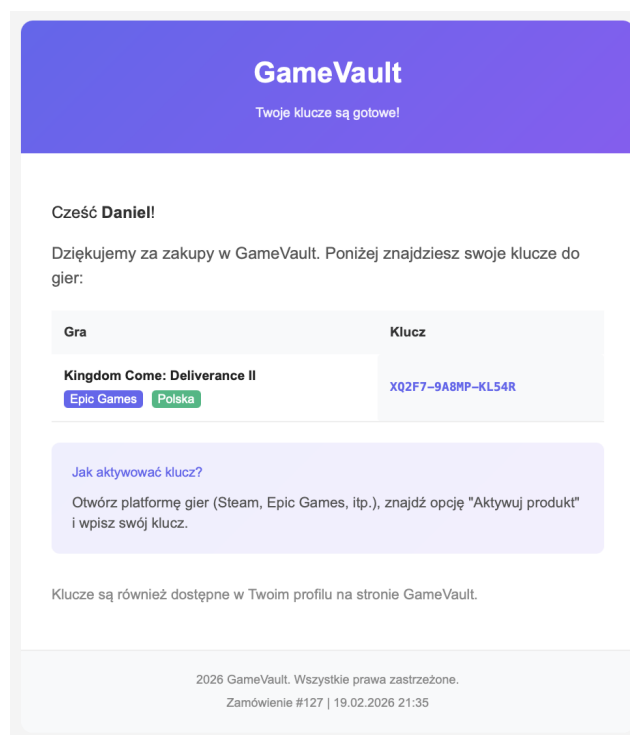
Rysunek 4.8: Okno powiadomienia o pomyślnym zakupie

Od tego momentu użytkownik może odebrać zakupione klucze w swoim profilu lub sprawdzić wiadomość e-mail, na którą zostały one wysłane (rys. 4.9, 4.10).

Moje klucze



Rysunek 4.9: Klucze klienta w jego profilu



Rysunek 4.10: Wygenerowany formularz na pocztę klienta z informacją o zakupie

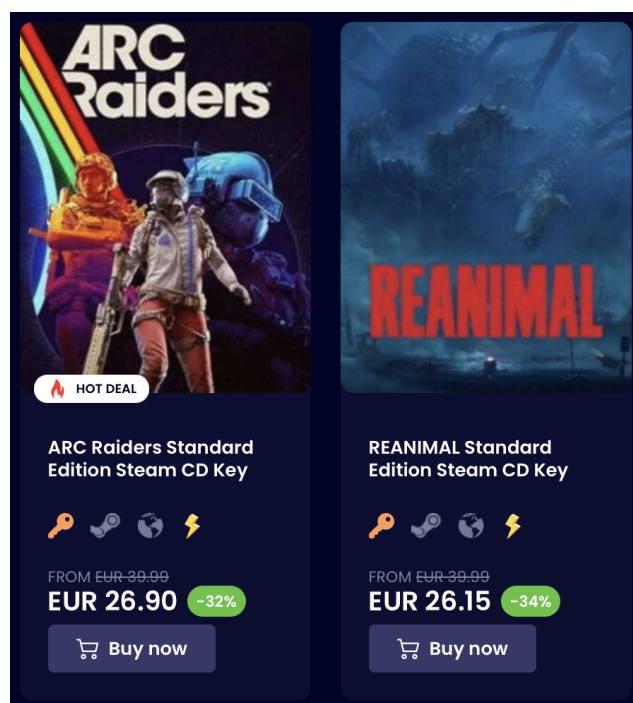
4.2 Ocena równoległej prezentacji ofert

Po przeprowadzeniu analizy konkurencyjnych platform, takich jak ENEBA, Kinguin oraz K4G, które działają w tej samej niszy rynkowej, mogą stwierdzić, że większość z nich stosuje równoległą prezentację ofert, wyświetlając je obok siebie (rys. 4.11, 4.12, 4.13). Rozwiązanie to umożliwia użytkownikowi szybkie porównanie najważniejszych informacji, takich jak region, platforma oraz cena. Taki sposób prezentacji skraca czas potrzebny na podjęcie decyzji zakupowej oraz zmniejsza ryzyko pomyłki przy wyborze oferty. Z perspektywy

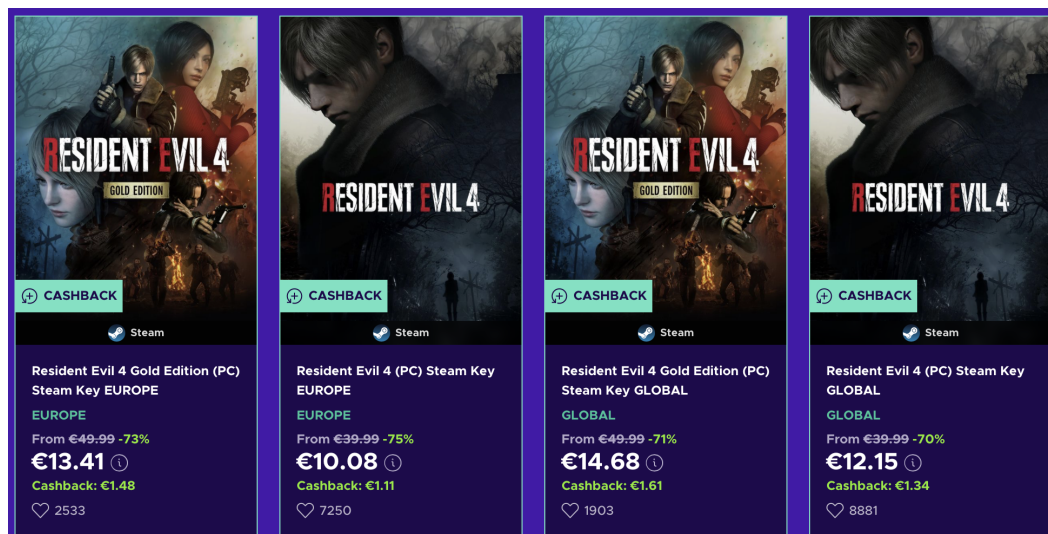
marketingowej jest to skuteczne rozwiązanie, ponieważ kluczowe informacje są widoczne od razu i łatwo dostępne dla klienta. Z tego powodu moja aplikacja korzysta z tej strategii.



Rysunek 4.11: Widok równoległej prezentacji ofert na konkurencyjnej platformie Kinguin [17]



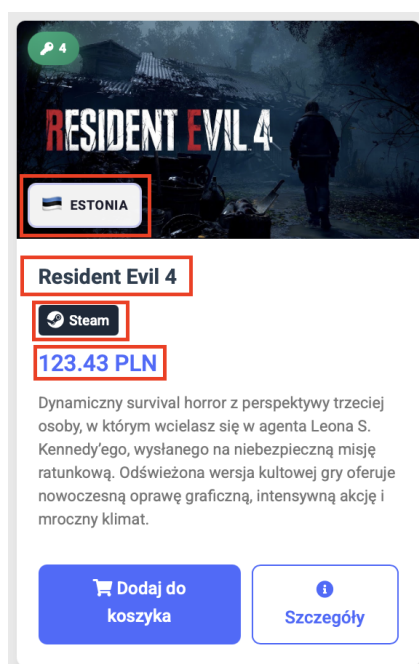
Rysunek 4.12: Widok równoległej prezentacji ofert na konkurencyjnej platformie K4G [15]



Rysunek 4.13: Widok równoległej prezentacji ofert na konkurencyjnej platformie ENEBA [4]

4.3 Najważniejsze atrybuty karty oferty

Na podstawie analizy konkurencyjnych platform wyciągnąłem wnioski, że najważniejszymi informacjami na karcie oferty są: tytuł gry, region, platforma oraz cena końcowa (rys. 4.14). To właśnie te elementy mają kluczowe znaczenie dla użytkownika podczas podejmowania decyzji zakupowej, ponieważ pozwalają szybko ocenić, czy dana oferta spełnia jego oczekiwania.



Rysunek 4.14: Najważniejsze atrybuty karty oferty

4.4 Test przejrzystości ceny końcowej

Na podstawie przeprowadzonych testów w trakcie tworzenia aplikacji podjąłem decyzję, że najlepszym rozwiązaniem będzie prezentowanie ceny końcowej już na karcie produktu, z uwzględnieniem wszystkich podatków. Takie podejście eliminuje sytuację, w której dodatkowe opłaty są doliczane dopiero na etapie koszyka. W moim systemie zdecydowałem się na pełną transparentność cen, aby ułatwić użytkownikowi porównywanie ofert oraz zwiększyć wygodę i przejrzystość procesu zakupowego.

4.5 Porównanie mojej platformy z oficjalnymi platformami

Kryterium	Moja platforma	Oficjalne sklepy (Steam, Epic Games)
Cena	Ceny są niższe dzięki modelowi sprzedaży kluczy. Przykład: <i>GTA V</i> za ok. 30 zł.	Ceny są zwykle wyższe. Przykład: <i>GTA V</i> za ok. 60 zł.
Porównanie ofert	Oferty dla różnych platform zebrane są w jednym miejscu, więc użytkownik może szybko porównać warianty bez opuszczania serwisu.	Aby znaleźć najtańszą opcję, użytkownik często musi przełączać się między sklepami (np. Steam i Epic Games) i porównywać ceny ręcznie.
Liczba kupowanych kopii	Użytkownik może kupić dowolną liczbę kluczy (np. na potrzeby prezentu lub dla znajomych).	Zwykle możliwy jest zakup jednej kopii gry na jedno konto/profil użytkownika.
Filtrowanie ofert	Dostępne są rozbudowane filtry umożliwiające wyszukiwanie gier według platformy oraz regionu. Użytkownik może dowolnie wybrać interesującą go kombinację parametrów.	Filtrowanie dotyczy wyłącznie danej platformy (np. Steam lub Epic Games). Region jest domyślnie przypisany do konta podczas rejestracji i nie ma możliwości swobodnego filtrowania według regionu.

Tabela 4.1: Porównanie mojej platformy GameVault z oficjalnymi sklepami cyfrowej dystrybucji

4.6 Ograniczenia wdrożeniowe projektu

Już na etapie tworzenia prototypu zacząłem poszukiwać partnerów, aby w przyszłości móc nawiązać z nimi współpracę. Udało mi się znaleźć potencjalnego partnera z Estonii, z którym w przyszłości będzie można rozpocząć oficjalną współpracę [2].

Ponieważ mieszkam na Litwie, muszę zarejestrować działalność gospodarczą, aby móc oficjalnie uruchomić i prowadzić tutaj swoją platformę. Na etapie pisania pracy dyplomowej nie miałem możliwości zrealizowania tych działań, ponieważ oficjalne otwarcie firmy na Litwie jest procesem czasochłonnym. Po drugie, bez otwarcia oficjalnej firmy nie jest możliwe pełne podłączenie systemu płatności poprzez partnera Stripe, dlatego w projekcie zostały wykorzystane jedynie testowe narzędzia płatności tego dostawcy.

Po trzecie, zarówno partner z Estonii, jak i Stripe wymagają, aby platforma była oficjalnie opublikowana w Internecie na własnej domenie [37]. Dopiero wtedy możliwe jest pełne korzystanie z ich usług. Dodatkowo znalazłem na przyszłość również niedrogi partnera hostingowego pod nazwą Hostinger [10].

Po czwarte, aby uruchomić platformę na Litwie, powinna ona być dostępna w języku litewskim lub przynajmniej angielskim. Ponieważ platforma na etapie prototypu została stworzona w języku polskim, wymaga pełnego przetłumaczenia przynajmniej na jeden z tych dwóch języków. W planach produkcyjnych na początkowym etapie można jednak zastosować prostsze rozwiązanie, polegające na pozostawieniu platformy w języku polskim i rozpoczęciu sprzedaży kluczy na rynku polskim. W takim przypadku nadal konieczna byłaby realizacja wszystkich wcześniej wymienionych punktów, z wyjątkiem tłumaczenia platformy na język litewski lub angielski.

Podsumowując tę część pracy, chciałbym podkreślić, że skupiłem się przede wszystkim na stworzeniu solidnego i działającego prototypu systemu, który w przyszłości będzie można rozwinąć i wdrożyć do pełnej, oficjalnej wersji produkcyjnej.

Podsumowanie

Celem niniejszej pracy było zaprojektowanie, zaimplementowanie oraz ocenienie prototypu aplikacji webowej GameVault, służącej do sprzedaży cyfrowych kluczy do gier. Głównym założeniem systemu było stworzenie miejsca, w którym użytkownik może szybko znaleźć interesujący go produkt, porównać dostępne warianty, sprawdzić najważniejsze informacje dotyczące platformy, regionu i ceny, a następnie przejść przez cały proces zakupu. W pracy szczególną uwagę zwróciłem na przejrzystość oferty, bezpieczeństwo działania aplikacji oraz wygodę użytkownika.

W ramach pracy udało się przygotować działający prototyp systemu, który obejmuje najważniejsze funkcje typowe dla sklepu internetowego z kluczami cyfrowymi. Zaimplementowano rejestrację i logowanie użytkowników, obsługę sesji, funkcję „Zapamiętaj mnie”, przeglądanie produktów, filtrowanie ofert, koszyk zakupowy, finalizację zamówienia, przypisywanie kluczy do konta użytkownika oraz historię zakupów. Dodatkowo przygotowano panel administratora, w którym możliwe jest zarządzanie produktami, kluczami, użytkownikami, kodami rabatowymi oraz zgłoszeniami technicznymi. Dzięki temu aplikacja nie ogranicza się wyłącznie do prostego widoku sklepu, ale zawiera również zaplecze administracyjne potrzebne do obsługi systemu.

Ważnym elementem pracy było zaprojektowanie bazy danych. Struktura bazy została podzielona na kilka modułów: użytkowników i produktów, zakupów oraz wsparcia technicznego. Zastosowano relacje między tabelami, klucze obce oraz odpowiednie typy danych, co pozwala zachować porządek i spójność informacji. W projekcie uwzględniono również podstawowe mechanizmy bezpieczeństwa, takie jak haszowanie haseł za pomocą silnych algorytmów `password_hash()`, weryfikację haseł przez `password_verify()`, stosowanie zapytań przygotowanych, zabezpieczenie danych wyświetlanych na stronie przez `htmlspecialchars()` oraz tokeny CSRF w formularzach administracyjnych. Dzięki temu system został przygotowany w sposób bardziej odporny na typowe zagrożenia, takie jak SQL Injection, XSS oraz nieautoryzowane wysyłanie formularzy.

Na podstawie wykonanych testów można stwierdzić, że przygotowany prototyp spełnia główne założenia pracy. Użytkownik może przejść przez standardowy proces zakupu: od wyszukania produktu, przez dodanie go do koszyka, aż do finalizacji zamówienia i odebrania klucza. Równoległa prezentacja ofert

ułatwia porównanie produktów, ponieważ najważniejsze informacje są widoczne od razu na karcie produktu. Dotyczy to przede wszystkim ceny, platformy aktywacji, regionu oraz wariantu produktu. Takie rozwiązanie zmniejsza liczbę kroków potrzebnych do podjęcia decyzji i ogranicza ryzyko pomyłki, na przykład zakupu klucza przeznaczonego na inną platformę lub region.

W pracy potwierdzono również, że jawna prezentacja ceny końcowej ma duże znaczenie dla wiarygodności systemu. Użytkownik od początku widzi kwotę, którą ma zapłacić, bez ukrytych opłat doliczanych na późniejszym etapie. Takie rozwiązanie zwiększa przejrzystość procesu zakupowego i ułatwia porównywanie ofert.

Należy jednak podkreślić, że przygotowany system jest prototypem, a nie pełną wersją produkcyjną. W pracy wykorzystano dane symulowane, ponieważ celem było przede wszystkim sprawdzenie koncepcji oraz funkcjonalności aplikacji. Pełne wdrożenie systemu wymagałoby nawiązania współpracy z legalnymi dostawcami kluczy, uruchomienia platformy na własnej domenie, przygotowania wersji językowej odpowiedniej dla wybranego rynku oraz pełnego skonfigurowania płatności produkcyjnych. W przypadku Stripe konieczne byłoby również spełnienie wymagań formalnych związanych z prowadzeniem działalności gospodarczej oraz obsługą płatności w środowisku produkcyjnym.

W przyszłości aplikację można rozwinąć o kolejne funkcje. Jednym z najważniejszych kierunków rozwoju byłaby integracja z rzeczywistymi dostawcami kluczy. Kolejnym krokiem mogłoby być dodanie rozbudowanego systemu opinii użytkowników, automatycznych powiadomień e-mail, zaawansowanego filtrowania ofert, obsługi wielu języków oraz pełnej wersji responsywnej dostosowanej do urządzeń mobilnych. Można byłoby również rozbudować panel administratora o szczegółowe statystyki sprzedaży, raporty finansowe i analizę popularności produktów.

Podsumowując, w ramach pracy udało się osiągnąć główny cel, którym było stworzenie działającego prototypu internetowej platformy sprzedaży kluczy do gier. System pokazuje, że możliwe jest przygotowanie przejrzystej aplikacji, która łączy funkcje sklepu internetowego, porównywarki ofert oraz panelu administracyjnego. Projekt może stanowić dobrą podstawę do dalszego rozwoju i w przyszłości zostać przekształcony w pełną platformę produkcyjną.

Spis rysunków

1.1	Logo platformy Steam	3
1.2	Logo platformy Epic Games	4
1.3	Logo platformy PlayStation Store	4
1.4	Logo platformy Xbox	5
2.1	Diagram BPMN	13
2.2	Diagram przypadków użycia	14
2.3	Diagram ERD struktury bazy danych	15
3.1	Formularz rejestracji	20
3.2	Formularz logowania	22
3.3	Przycisk „Wyloguj”	24
3.4	Dodanie produktu do koszyka poprzez niezalogowaną osobę . .	26
3.5	Komunikat ostrzegawczy jeżeli osoba niezalogowana próbuje dodać produkt do koszyka	27
3.6	Jeśli produkt istnieje w koszyku, to przy ponownym jego doda- niu zwiększa się ilość o jeden	27
3.7	Przekierowanie do koszyka	27
3.8	Usunięcie produktu z koszyka	28
3.9	Główne okno panelu administracyjnego	29
3.10	Lista produktów w zakładce „Produkty” w panelu administratora	30
3.11	Pasek wyszukiwania w zakładce „Produkty” w panelu admini- stratora	31
3.12	Pusty formularz w panelu administratora	33
3.13	Stan magazynowy kluczy w panelu administratora	38
3.14	Dodanie nowego klucza poprzez panel administratora	39
3.15	Lista dodanych kluczy poprzez panel administratora	41
3.16	Tabela z kodami promocyjnymi w panelu administratora . . .	42
3.17	Pole dodania nowego kodu rabatowego w panelu administratora	43
3.18	Główne okno systemu zgłoszeń od strony administratora . . .	45
3.19	Okno chatu w systemie zgłoszeń od strony administratora . .	46
4.1	Pole do wyszukiwania produktów z filtrami na stronie głównej	49
4.2	Karty gier obok siebie	49
4.3	Pierwsza część widoku szczegółów gry	50

4.4	Druga część widoku szczegółów gry	51
4.5	Widok koszyka	52
4.6	Okno finalizacji zakupu	52
4.7	Formularz opłaty po stronie partnera	53
4.8	Okno powiadomienia o pomyślnym zakupie	53
4.9	Klucze klienta w jego profilu	54
4.10	Wygenerowany formularz na pocztę klienta z informacją o zakupie	54
4.11	Widok równoległej prezentacji ofert na konkurencyjnej platformie Kinguin	55
4.12	Widok równoległej prezentacji ofert na konkurencyjnej platformie K4G	55
4.13	Widok równoległej prezentacji ofert na konkurencyjnej platformie ENEBA	56
4.14	Najważniejsze atrybuty karty oferty	56

Spis tabel

2.1	Wymagania funkcjonalne systemu GameVault	11
2.2	Wymagania niefunkcjonalne systemu GameVault	12
2.3	Model bazy danych	16
2.4	Relacje między tabelami – klucze obce (FK)	18
4.1	Porównanie mojej platformy GameVault z oficjalnymi sklepami cyfrowej dystrybucji	57

Bibliografia

- [1] Apache Friends – *XAMPP*,
www.apachefriends.org
Dostęp na dzień 2025.10.21
- [2] DIGITAL-CODES – *partner z Estonii*,
<https://www.digitalcodeswholesale.com>
Dostęp na dzień 2026.03.12
- [3] Eneba – *About us*,
www.eneba.com/about-us?srsltid=AfmB0or6TwMRK-JniJT2baIG_GLWT_AejNTGiR2tjZgf3U182wr8fuq2X
Dostęp na dzień 2025.10.21
- [4] Eneba – *strona główna platformy*,
<https://www.eneba.com>
Dostęp na dzień 2026.05.20
- [5] Eneba – *Sell on Eneba (panel, rozliczenia, API)*,
www.eneba.com/us/sell-with-eneba?srsltid=AfmB0oot2QCld1bs2_zhZyo_kjtbkF23B2ClFZtxXhicBA3Ff92R8Kuq
Dostęp na dzień 2025.10.21
- [6] Eneba – *strona główna, charakter oferty*,
www.eneba.com/us/?srsltid=AfmB0opGtuH8XZWkiSEr-qwXZLFtsz4vrIhAPShplfWnyJV_Oscbrj5T
Dostęp na dzień 2025.10.21
- [7] Epic Games – *Redeem your gift card*,
www.epicgames.com/redeem/en-US
Dostęp na dzień 2025.10.20
- [8] Epic Games Help – *How do I redeem a code to my Epic Games account?*,
<https://www.epicgames.com/help/en-US/epic-games-store-c-20230000001639/launcher-support-c-202300000001735/how-do-i-redeem-a-code-to-my-epic-games-account-a202300000012762>
Dostęp na dzień 2025.10.20

-
- [9] Epic Games Store – *Redeem Code*,
<http://store.epicgames.com/en-US/redeem>
Dostęp na dzień 2025.10.13
 - [10] Hostinger – *partner dla zamieszczenia mojej platformy na stronę internetową*,
<https://www.hostinger.com>
Dostęp na dzień 2026.03.12
 - [11] Icon-Icons – *Epic Games black logo*,
<https://icon-icons.com/icon/epic-games-black-logo/147139>
Dostęp na dzień 2026.05.20
 - [12] Logo.wine – *Xbox app logo*,
https://www.logo.wine/logo/Xbox_%28app%29
Dostęp na dzień 2026.05.20
 - [13] K4G – *Start Listing (formularz dla sprzedawców)*,
<https://help.k4g.com/suppliers>
Dostęp na dzień 2025.10.21
 - [14] K4G Help Center – *Suppliers (how to start listing/become a supplier)*,
<https://help.k4g.com/suppliers>
Dostęp na dzień 2025.10.21
 - [15] K4G – *strona główna platformy*,
<https://k4g.com>
Dostęp na dzień 2026.05.20
 - [16] Kinguin – *Legal Notice (obowiązki sprzedawców, uprawnienia/EULA)*,
www.kinguin.net/page/legal
Dostęp na dzień 2025.10.21
 - [17] Kinguin – *strona główna platformy*,
<https://www.kinguin.net>
Dostęp na dzień 2026.05.20
 - [18] Kinguin – *Terms and Conditions (rola administratora serwisu; Users-Sellers)*,
www.kinguin.net/integration/terms-and-conditions
Dostęp na dzień 2025.10.21
 - [19] MDN Web Docs – *CSS: Cascading Style Sheets*,
<https://developer.mozilla.org/en-US/docs/Web/CSS>
Dostęp na dzień 2025.10.21

-
- [20] MDN Web Docs – *HTML: HyperText Markup Language*,
<https://developer.mozilla.org/en-US/docs/Web/HTML>
Dostęp na dzień 2025.10.21
- [21] MDN Web Docs – *JavaScript*,
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>
Dostęp na dzień 2026.02.04
- [22] MySQL Manual – *What is MySQL*,
<https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.html>
Dostęp na dzień 2025.10.21
- [23] PC Gamer – *Report shows 90 percent of European gaming revenue in 2024 was digital purchases...* (trend przejścia z fizycznych wydań na cyfrowe),
<https://www.pcgamer.com/gaming-industry/report-shows-90-percent-of-european-gaming-revenue-in-2024-was-digital-purchases-but-only-15-percent-was-on-pc>
Dostęp na dzień 2025.10.13
- [24] PHP Manual – *dokumentacja*,
<https://www.php.net/manual/en/index.php>
Dostęp na dzień 2025.10.21
- [25] PlayStation App – *Remote downloads & zarządzanie biblioteką*,
<https://www.playstation.com/en-us/support/games/playstation-app-features>
Dostęp na dzień 2025.10.20
- [26] PlayStation – *console-only games*,
<https://www.playstation.com/en-us/support/games/playstation-app-features>
Dostęp na dzień 2025.10.23
- [27] PlayStationPlus,
<https://www.playstation.com/en-us/ps-plus>
Dostęp na dzień 2025.10.23
- [28] PlayStation Support – *Redeem a PlayStation Store voucher code*,
www.playstation.com/en-us/support/store/redeem-ps-store-voucher-code
Dostęp na dzień 2025.10.20
- [29] SeekLogo – *PlayStation Store vector logo*,
<https://seeklogo.com/vector-logo/527764/playstation-store>
Dostęp na dzień 2026.05.20

-
- [30] Steam – *Gabe Newell*,
https://en.wikipedia.org/wiki/Gabe_Newell
Dostęp na dzień 2025.10.23
- [31] Steam Support – *Activating a Product on Steam*,
<https://help.steampowered.com/en/faqs/view/2A12-9D79-C3D7-F870>
Dostęp na dzień 2025.10.13
- [32] Steam Support – *Steam Families: User Guide*,
<https://help.steampowered.com/en/faqs/view/054C-3167-DD7F-49D4>
Dostęp na dzień 2025.10.20
- [33] Steam Support – *Revoked CD Keys*,
<https://help.steampowered.com/en/faqs/view/029B-6D58-6EE6-1D7A>
Dostęp na dzień 2025.10.13
- [34] Steam Support – *Region Restrictions on Steam*,
<https://help.steampowered.com/en/faqs/view/58D3-B80D-2943-3CC6>
Dostęp na dzień 2025.10.13
- [35] Steamworks Docs – *Steam Cloud*,
<https://partner.steamgames.com/doc/features/cloud>
Dostęp na dzień 2025.10.20
- [36] Steamworks Documentation – *Steam Keys*,
<https://partner.steamgames.com/doc/features/keys>
Dostęp na dzień 2025.10.13
- [37] Stripe – *partner dla realizacji opłaty*,
<https://stripe.com>
Dostęp na dzień 2026.03.12
- [38] Steam – *Tomb Raider Game of the Year Edition*,
https://store.steampowered.com/app/203160/Tomb_Raider_Game_of_the_Year/
Dostęp na dzień 2026.04.04
- [39] Steam – *przykładowe DLC do gry*,
https://store.steampowered.com/app/2778580/ELDEN_RING_Shadow_of_the_Erdtree/
Dostęp na dzień 2026.04.04

- [40] TechRadar – *Physical software accounted for 3 percent of PlayStation sales...* (aktualny wskaźnik marginalizacji sprzedaży fizycznej),
www.techradar.com/gaming/sony-reveals-that-physical-software-only-accounted-for-3-percent-of-playstation-sales-in-the-last-year-only-further-fuelling-the-fire-that-the-ps6-will-continue-the-embrace-of-digital-gaming
Dostęp na dzień 2025.10.13
- [41] The Guardian – *Boxed video game sales collapse in UK as digital revenues flatten* (spadki sprzedaży pudełkowej, wzrost udziału cyfry),
<https://www.theguardian.com/games/2025/jan/08/boxed-video-game-sales-collapse-in-uk-as-digital-revenues-flatten>
Dostęp na dzień 2025.10.13
- [42] The Verge – *Steam's improved family sharing is out now for everyone*,
www.theverge.com/2024/9/11/24242377/valve-steam-improved-family-sharing-out-now
Dostęp na dzień 2025.10.20
- [43] Wikimedia Commons – *Steam icon logo*,
https://commons.wikimedia.org/wiki/File:Steam_icon_logo.svg
Dostęp na dzień 2026.05.20
- [44] Wikipedia – *Steam (service)*,
[https://en.wikipedia.org/wiki/Steam_\(service\)](https://en.wikipedia.org/wiki/Steam_(service))
Dostęp na dzień 2025.10.23
- [45] Wikipedia – *Downloadable content*,
https://en.wikipedia.org/wiki/Downloadable_content
Dostęp na dzień 2026.04.04
- [46] Xbox – *Why Digital Games*,
www.xbox.com/en-US/microsoft-store/why-digital-games
Dostęp na dzień 2025.10.21
- [47] Xbox.com – *Xbox PC app*,
www.xbox.com/en-US/apps/xbox-app-on-pc
Dostęp na dzień 2025.10.21
- [48] Xbox Support – *How to redeem gift cards and codes*,
support.xbox.com/en-US/help/subscriptions-billing/redeem-codes-gifting/redeem-prepaid-codes
Dostęp na dzień 2025.10.20