WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: Programowanie

Patryk Rzegocki
Nr albumu studenta w69840

Gra fabularna przeglądarkowa RPG

Prowadzący: mgr inż. Łukasz Piechocki

Praca projektowa Szkolenie techniczne 2

Rzeszów 2025

Spis treści

Wstęp

W dzisiejszym świecie, w którym technologia i rozrywka odgrywają kluczową rolę w naszym życiu, wiele osób poszukuje sposobów na oderwanie się od codziennych obowiązków i stresów. Gry RPG maja na celu nie tylko dostarczenie rozrywki, ale także stworzenie unikalnej przestrzeni, w której ludzie mogą przeżywać przygody, rozwijać swoje umiejętności i zanurzać się w wirtualnym świecie, praktykując eskapizm. W świecie gry, gracze będą mieli możliwość wcielenia się w różnorodne postacie, które będą musiały stawić czoła wyzwaniom, zagadkom i potworom. Szczególnie przyciągająca jest wyjątkowa forma - niespotykany sposób przedstawienia gry i wciągające mechaniki potrafią zapewnić ludziom godziny zabawy. Z tego powodu, ważne jest, by znaleźć rozwiązanie na powyższe zagadnienie, co ten projekt stara się zrealizować.

Rozdział 1

Opis zastosowanego stosu technologicznego i narzędzi

Projekt "Shakes & Godmist"wykorzystuje nowoczesny stos technologiczny, który obejmuje zarówno technologie backendowe, jak i frontendowe oraz narzędzia wspierające proces wytwarzania oprogramowania.

1.1 Języki programowania

- **C#** – główny język backendu, wykorzystywany w ASP.NET Core.
- **TypeScript** – główny język frontendowy, wykorzystywany w Angularze.
- **SQL** – do obsługi bazy danych PostgreSQL.
- **HTML, CSS** – do budowy interfejsu użytkownika.

1.2 Frameworki i biblioteki

- **ASP.NET Core 9.0** – framework do budowy REST API, obsługa routingu, autoryzacji, walidacji i logiki biznesowej.
- **Entity Framework Core 9.0.6** – ORM do mapowania obiektowo-relacyjnego i migracji bazy danych.
- **Angular 20.x** – framework frontendowy do budowy SPA, obsługa routingu, komponentów, serwisów i reaktywności.
- **RxJS 7.8.0** – biblioteka do reaktywnego zarządzania stanem i asynchronicznością w Angularze.
- **Swagger / Swashbuckle 9.0.1** – automatyczna dokumentacja i testowanie API.
- **JWT (JSON Web Token)** – mechanizm autoryzacji i uwierzytelniania użytkowników.
- **PostgreSQL** – relacyjna baza danych, przechowująca dane graczy, przedmiotów, misji itp.
- **Npgsql 9.0.3** – sterownik .NET do komunikacji z PostgreSQL.
- **Node.js** – środowisko uruchomieniowe dla Angular CLI i narzędzi frontendowych.
- **Angular CLI 20.x** – narzędzie do generowania, budowania i testowania aplikacji Angular.
- **TypeScript 5.8.2** – język programowania dla Angulara.

- **Zone.js 0.15.0** – obsługa stref asynchronicznych w Angularze.
- **Karma, Jasmine** – narzędzia do testów jednostkowych w Angularze.
- **Postman** – narzędzie do testowania i automatyzacji zapytań HTTP do API.
- **Git** – system kontroli wersji.
- **Visual Studio, JetBrains Rider, VS Code** – środowiska IDE wykorzystywane podczas rozwoju.

1.3 Komunikacja frontend-backend

Aplikacja korzysta z architektury klient-serwer. Komunikacja odbywa się przez REST API (JSON) na endpointach udostępnianych przez ASP.NET Core. Autoryzacja użytkownika realizowana jest przez tokeny JWT przesyłane w nagłówkach HTTP.

1.4 Zarządzanie stanem i bezpieczeństwo

- Po stronie backendu: autoryzacja i uwierzytelnianie użytkowników, walidacja danych, logika biznesowa (generowanie misji, walki, nagrody, sklep, ekwipunek).
- Po stronie frontendu: zarządzanie stanem gracza, obsługa sesji, prezentacja danych, obsługa błędów i komunikatów.
- Dane wrażliwe (hasła) są haszowane i nie są przechowywane w postaci jawnej.

1.5 Wersje kluczowych technologii

- ASP.NET Core: 9.0
- Entity Framework Core: 9.0.6
- Angular: 20.x
- Angular CLI: 20.x
- RxJS: 7.8.0
- TypeScript: 5.8.2
- Zone.js: 0.15.0
- Npgsql: 9.0.3
- Swashbuckle: 9.0.1
- Node.js: 18.x lub wyższy
- PostgreSQL: 14.x lub wyższy (zalecane)

1.6 Inne narzędzia i dobre praktyki

- **Swagger UI** – interaktywna dokumentacja API dostępna pod endpointem /swagger.
- **Migrations** – migracje bazy danych zarządzane przez EF Core.
- **Testy jednostkowe** – przykładowe testy dla Angulara (Karma/Jasmine).
- **Linting i formatowanie** – narzędzia do automatycznego sprawdzania i poprawy stylu kodu.
- **Responsywny interfejs** – CSS i Angular zapewniają poprawne wyświetlanie na różnych urządzeniach.

Rozdział 2

Instrukcja uruchomienia aplikacji

Aby uruchomić projekt "Shakes & Godmist"na swoim komputerze, należy kroki opisane w tym rozdziale.

Uwaga: Aplikacja została zaprojektowana z myślą o wdrożeniu na serwerze (np. VPS, chmura, serwer dedykowany). Uruchamianie lokalne służy wyłącznie celom deweloperskim i testowym. W środowisku produkcyjnym zaleca się wdrożenie backendu oraz bazy danych na serwerze, a frontend na serwerze WWW lub w usłudze hostingowej.

2.1 Wymagania systemowe

- **System operacyjny:** Windows 10/11, Linux lub macOS
- **.NET SDK:** wersja 9.0 lub wyższa
- **Node.js:** wersja 18.x lub wyższa
- **Angular CLI:** wersja 20.x
- **PostgreSQL:** wersja 14.x lub wyższa
- **RAM:** minimum 4 GB (zalecane 8 GB)
- **Dysk:** minimum 500 MB wolnego miejsca

2.2 Instalacja zależności

Backend (.NET):

1. Przejdź do katalogu backendu:

```
cd GameBackend
```

2. Przywróć zależności NuGet:

```
dotnet restore
```

Frontend (Angular):

1. Przejdź do katalogu frontendu:

```
cd GameFrontend
```

2. Zainstaluj zależności npm:

```
npm install
```

2.3 Konfiguracja bazy danych

1. Zainstaluj i uruchom serwer PostgreSQL.
2. Utwórz nową bazę danych, np. o nazwie `shakesgodmist`.
3. Skonfiguruj połączenie w pliku `appsettings.json` w katalogu GameBackend, np.:

```
"ConnectionStrings": {
  "DefaultConnection": "Host=localhost;Port=5432;
  Database=shakesgodmist;
  Username=postgres;Password=twoje_haslo"
}
```

2.4 Uruchamianie backendu

1. Wykonaj migracje bazy danych:

```
dotnet ef database update
```

2. Uruchom serwer backendu:

```
dotnet run
```

3. Domyślnie backend będzie dostępny pod adresem: `http://localhost:5166`

2.5 Uruchamianie frontendu

1. W nowym terminalu przejdź do katalogu GameFrontend:

```
cd GameFrontend
```

2. Uruchom serwer deweloperski Angular:

```
ng serve
```

3. Frontend będzie dostępny pod adresem: `http://localhost:4200`

2.6 Dostęp do aplikacji

- **Panel użytkownika:** `http://localhost:4200`
- **API backendu:** `http://localhost:5166/api`
- **Swagger (dokumentacja API):** `http://localhost:5166/swagger`

2.7 Najczęstsze problemy i wskazówki

- Jeśli port 5166 lub 4200 jest zajęty, zmień konfigurację w plikach projektu lub uruchom na innym porcie.
- W przypadku błędów połączenia z bazą danych sprawdź poprawność connection string i czy serwer PostgreSQL działa.
- Jeśli pojawią się błędy przy migracjach, upewnij się, że masz zainstalowane narzędzia Entity Framework Core CLI.
- W przypadku problemów z zależnościami npm uruchom `npm install` ponownie.
- W razie problemów z uprawnieniami uruchom terminal jako administrator.

Rozdział 3

Diagram bazy danych

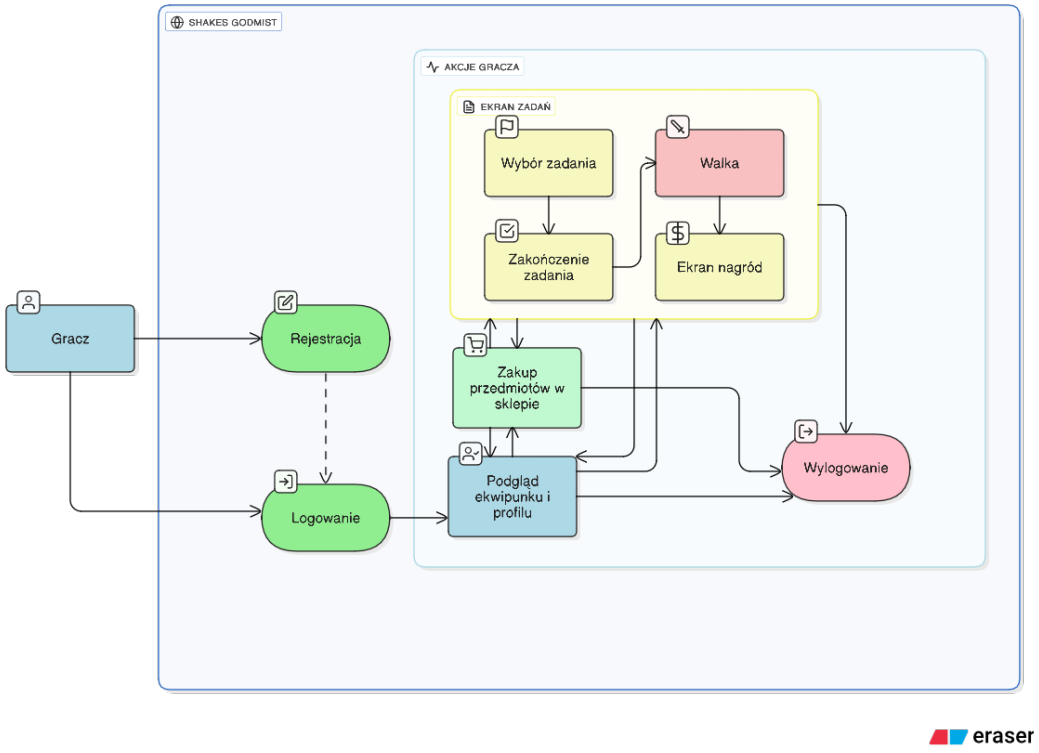
Na rysunku 3.1 znajduje się diagram bazy danych projektu, przedstawiający tabele oraz relacje pomiędzy nimi.

Rysunek 3.1: Diagram bazy danych

Rozdział 4

Diagram przypadków użycia UML

Na rysunku 3.2 przedstawiono diagram przypadków użycia systemu, ilustrujący główne interakcje użytkownika (gracza) z aplikacją.



Rysunek 4.1: Diagram przypadków użycia UML

Rozdział 5

Opis kluczowych elementów back-endu

W tej sekcji opisano najważniejsze elementy warstwy back-endowej projektu, ich odpowiedzialności oraz sposób działania.

- **GameDbContext** – Główny kontekst bazy danych, odpowiada za połączenie z bazą PostgreSQL oraz mapowanie encji (Player, Item, Quest, User, Enemy) na tabele w bazie. Umożliwia wykonywanie operacji CRUD na wszystkich obiektach domenowych. Konfiguracja relacji i kluczy obcych odbywa się w metodzie `OnModelCreating`. DbContext jest wykorzystywany przez wszystkie kontrolery do komunikacji z bazą danych.
- **AuthController** – Odpowiada za obsługę rejestracji, logowania i uwierzytelniania użytkowników. Główne metody:
  - `Register` – rejestracja nowego użytkownika, walidacja danych, wywołanie serwisu AuthService.
  - `Login` – logowanie użytkownika, sprawdzenie poprawności hasła, generowanie tokenu JWT.
  - `GetCurrentUser` – pobranie danych aktualnie zalogowanego użytkownika na podstawie tokenu.

  Kontroler komunikuje się z serwisem AuthService i zwraca odpowiedzi HTTP.
- **ItemsController** – Zarządza operacjami na przedmiotach gracza. Główne metody:
  - `GetItems` – pobiera listę przedmiotów gracza.
  - `BuyItem` – umożliwia zakup nowego przedmiotu przez gracza, sprawdza saldo i dostępność.
  - `SellItem` – pozwala sprzedać przedmiot z ekwipunku gracza.
  - `EquipItem` – przypisuje przedmiot do slotu ekwipunku gracza.

  Każda metoda waliduje uprawnienia użytkownika i wykonuje operacje na bazie przez GameDbContext.
- **PlayersController** – Obsługuje operacje związane z graczem. Główne metody:
  - `GetPlayer` – pobiera profil i statystyki gracza.
  - `UpgradeStats` – umożliwia ulepszanie statystyk gracza, sprawdza koszty i limity.
  - `GetInventory` – pobiera ekwipunek gracza.
  - `UpdatePlayer` – aktualizuje dane gracza (np. po walce lub misji).

  Kontroler sprawdza uprawnienia, pobiera i aktualizuje dane gracza w bazie, wywołuje logikę biznesową.

- **QuestsController** – Odpowiada za generowanie, akceptowanie i kończenie misji. Główne metody:
  - `GenerateQuests` – generuje nowe propozycje misji na podstawie poziomu gracza.
  - `AcceptQuest` – przypisuje wybraną misję do gracza, serializuje przeciwnika i nagrodę.
  - `CompleteQuest` – kończy misję, przyznaje nagrody, aktualizuje postęp gracza.
  - `GetActiveQuests` – pobiera aktualne misje gracza.

  Kontroler waliduje dostępność misji, uprawnienia gracza i zapisuje postęp w bazie.
- **AuthService** – Serwis odpowiedzialny za logikę uwierzytelniania. Główne metody:
  - `RegisterUser` – tworzy nowego użytkownika, hashuje hasło, zapisuje w bazie.
  - `ValidateUser` – sprawdza poprawność danych logowania.
  - `GenerateJwtToken` – generuje token JWT dla zalogowanego użytkownika.

  Oddziela logikę biznesową od kontrolera, zapewnia bezpieczeństwo i enkapsulację operacji na użytkownikach.
- **Podział odpowiedzialności** – Logika biznesowa (np. generowanie misji, walidacja, przetwarzanie nagród) znajduje się po stronie back-endu, natomiast kontrolery odpowiadają za obsługę żądań HTTP i komunikację z klientem. Dane są przechowywane w bazie PostgreSQL, a dostęp do nich realizowany jest przez Entity Framework Core. Każdy kontroler odpowiada za walidację uprawnień i poprawności danych wejściowych.

Rozdział 6

Przypadki testowe

W tej sekcji opisano przykładowe przypadki testowe dla kluczowych funkcjonalności systemu, zarówno po stronie backendu (API), jak i frontendu (interfejs użytkownika).

6.1 Testy backendu (API)

- **Rejestracja nowego użytkownika**
  - **Cel:** Sprawdzenie poprawności rejestracji użytkownika przez API.
  - **Warunki początkowe:** Brak użytkownika o podanym e-mailu w bazie.
  - **Kroki testowe:**
    1. Wysłanie żądania POST /api/auth/register z danymi użytkownika.
  - **Dane wejściowe:** Przykładowy e-mail, hasło, nazwa użytkownika.
  - **Oczekiwany rezultat:** Odpowiedź 200 OK, utworzenie nowego użytkownika w bazie.
  - **Wynik testu:**

Rysunek 6.1: Test rejestracji użytkownika

- **Logowanie użytkownika**
  - **Cel:** Sprawdzenie poprawności logowania i generowania tokenu JWT.
  - **Warunki początkowe:** Istnieje użytkownik z podanym e-mailem i hasłem.
  - **Kroki testowe:**

1. Wysłanie żądania POST /api/auth/login z poprawnymi danymi.

– **Dane wejściowe:** E-mail, hasło.
– **Oczekiwany rezultat:** Odpowiedź 200 OK, zwrócony token JWT.
– **Wynik testu:**

Rysunek 6.2: Test logowania użytkownika

- **Pobranie profilu gracza**

– **Cel:** Sprawdzenie możliwości pobrania danych gracza.
– **Warunki początkowe:** Użytkownik jest zalogowany (ma token JWT).
– **Kroki testowe:**

1. Wysłanie żądania GET /api/players z nagłówkiem Authorization: Bearer <token>.

– **Dane wejściowe:** Token JWT.
– **Oczekiwany rezultat:** Odpowiedź 200 OK, zwrócone dane gracza.
– **Wynik testu:**

Rysunek 6.3: Test danych gracza

- **Zakup przedmiotu**

– **Cel:** Sprawdzenie możliwości zakupu przedmiotu przez API.
– **Warunki początkowe:** Gracz jest zalogowany, ma wystarczającą ilość złota.
– **Kroki testowe:**

1. Wysłanie żądania POST /api/items/buy z danymi przedmiotu.

– **Dane wejściowe:** Identyfikator przedmiotu, token JWT.
– **Oczekiwany rezultat:** Odpowiedź 400 Bad Request, gracz nie ma wystarczającej ilości złota.

– **Wynik testu:**

Rysunek 6.4: Test zakupu przedmiotu - brak złota

- **Rozpoczęcie nowej misji**
  - **Cel:** Sprawdzenie możliwości rozpoczęcia misji przez API.
  - **Warunki początkowe:** Gracz jest zalogowany, ma dostępne misje.
  - **Kroki testowe:**
    1. Wysłanie żądania POST /api/quests/start z danymi misji.
  - **Dane wejściowe:** Identyfikator misji, token JWT.
  - **Oczekiwany rezultat:** Odpowiedź 200 OK, misja przypisana do gracza.
  - **Wynik testu:**

Rysunek 6.5: Test rozpoczęcia misji

- **Zakończenie misji**
  - **Cel:** Sprawdzenie poprawności zakończenia misji i przyznania nagrody.
  - **Warunki początkowe:** Gracz ma aktywną misję.
  - **Kroki testowe:**
    1. Wysłanie żądania POST /api/quests/complete z danymi misji.
  - **Dane wejściowe:** Identyfikator misji, token JWT.
  - **Oczekiwany rezultat:** Odpowiedź 200 OK, nagroda dodana do ekwipunku, XP i złoto zaktualizowane.
  - **Wynik testu:**

Rysunek 6.6: Test zakończenia misji

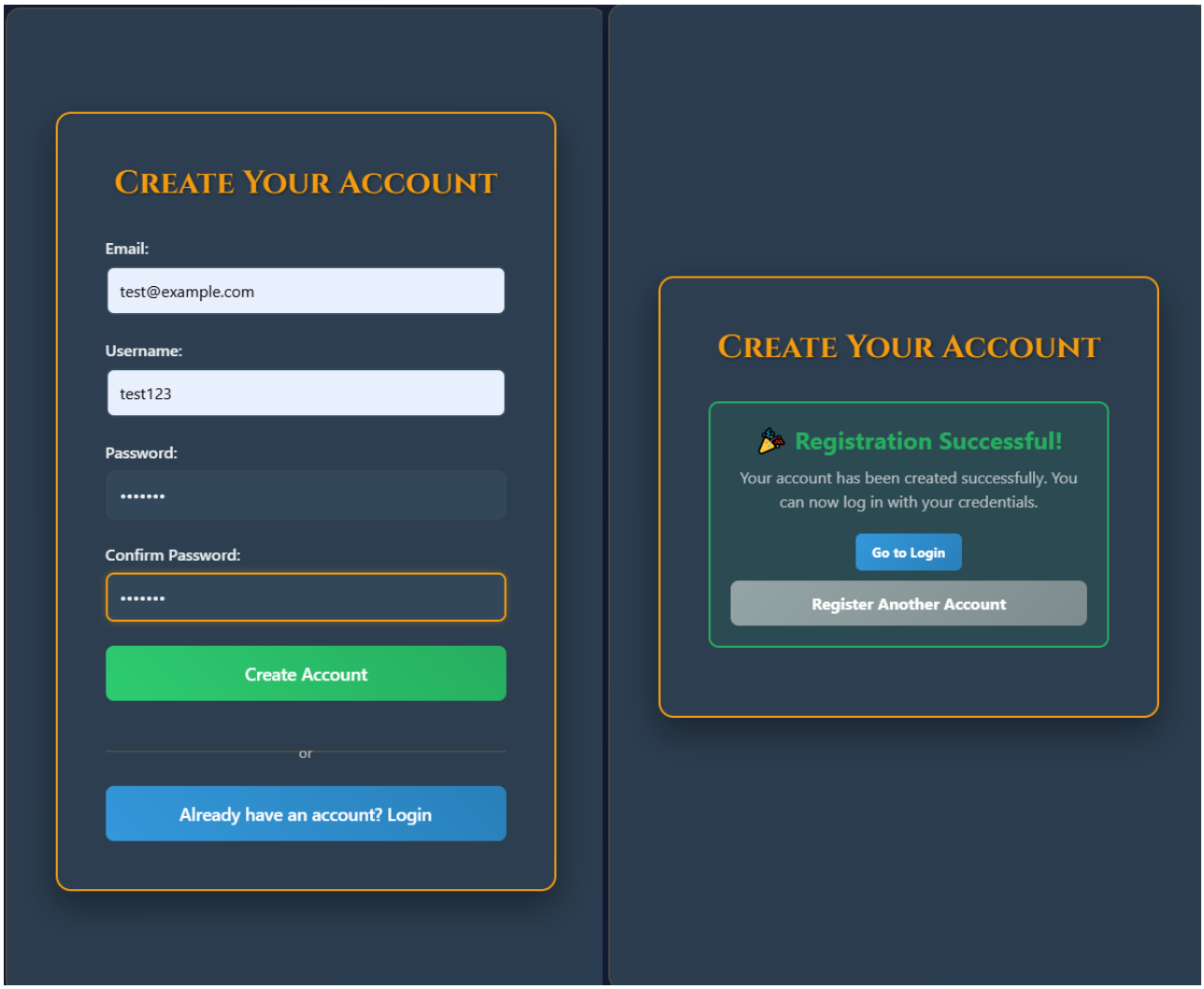
- **Obsługa nieautoryzowanego dostępu**
  - **Cel:** Sprawdzenie, czy API odrzuca żądania bez ważnego tokenu.
  - **Warunki początkowe:** Brak tokenu lub token nieprawidłowy.
  - **Kroki testowe:**
    1. Wysłanie żądania GET /api/players/me bez nagłówka Authorization.
  - **Dane wejściowe:** Brak lub nieprawidłowy token.
  - **Oczekiwany rezultat:** Odpowiedź 401 Unauthorized.
  - **Wynik testu:**

Rysunek 6.7: Test nieautoryzowanego dostępu

6.2 Testy frontendu (UI)

- **Rejestracja użytkownika przez interfejs**
  - **Cel:** Sprawdzenie poprawności działania formularza rejestracji.
  - **Warunki początkowe:** Brak zalogowanego użytkownika.
  - **Kroki testowe:**
    1. Otwórz stronę rejestracji.
    2. Wprowadź dane użytkownika i zatwierdź formularz.
  - **Dane wejściowe:** E-mail, hasło, nazwa użytkownika.
  - **Oczekiwany rezultat:** Komunikat o sukcesie, przekierowanie do ekranu logowania lub gry.

– **Wynik testu:**



Rysunek 6.8: Test rejestracji użytkownika - frontend

- **Logowanie użytkownika przez interfejs**

  – **Cel:** Sprawdzenie poprawności działania formularza logowania.

  – **Warunki początkowe:** Istnieje zarejestrowany użytkownik.

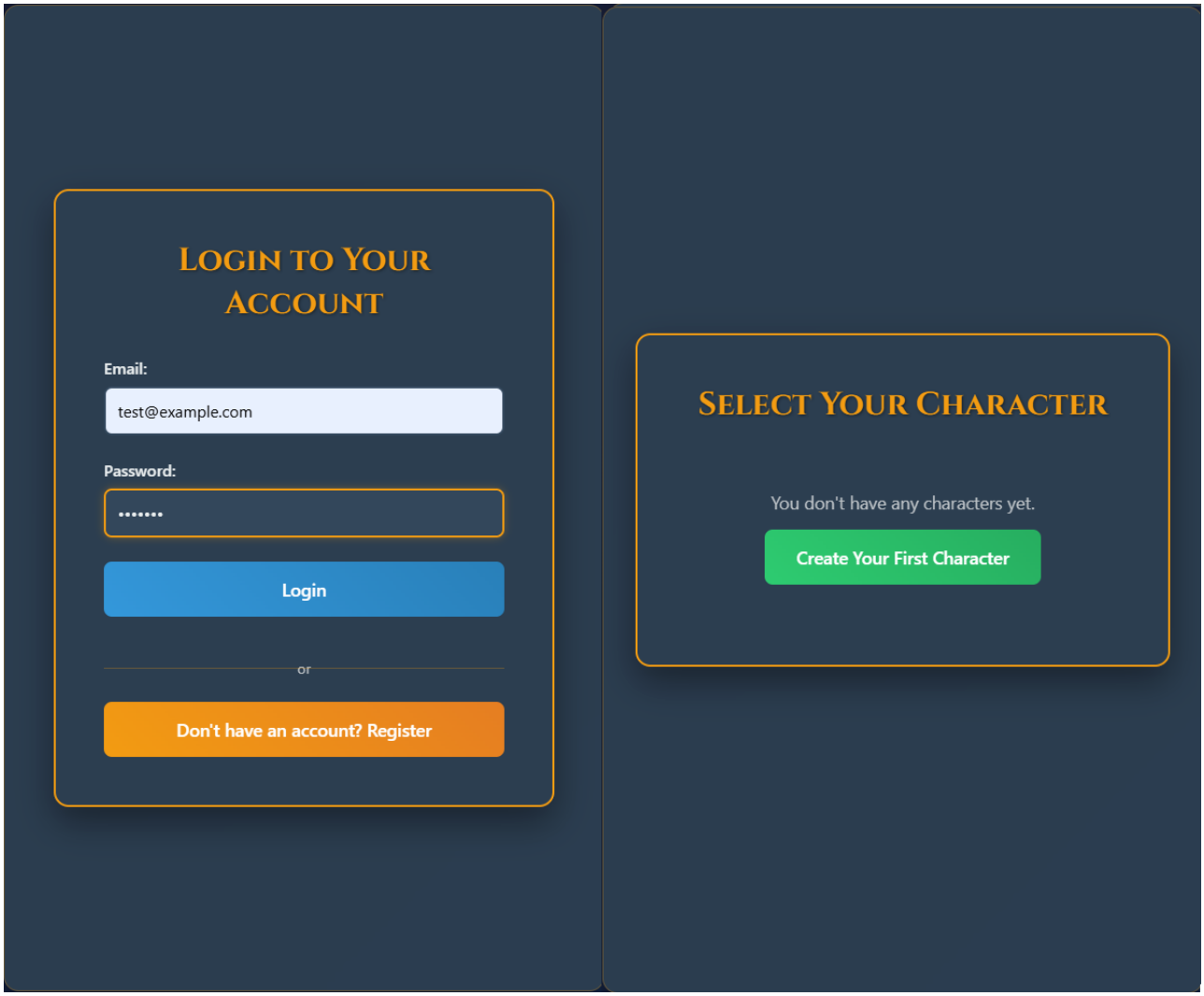
  – **Kroki testowe:**

    1. Otwórz stronę logowania.
    2. Wprowadź poprawne dane i zatwierdź formularz.

  – **Dane wejściowe:** E-mail, hasło.

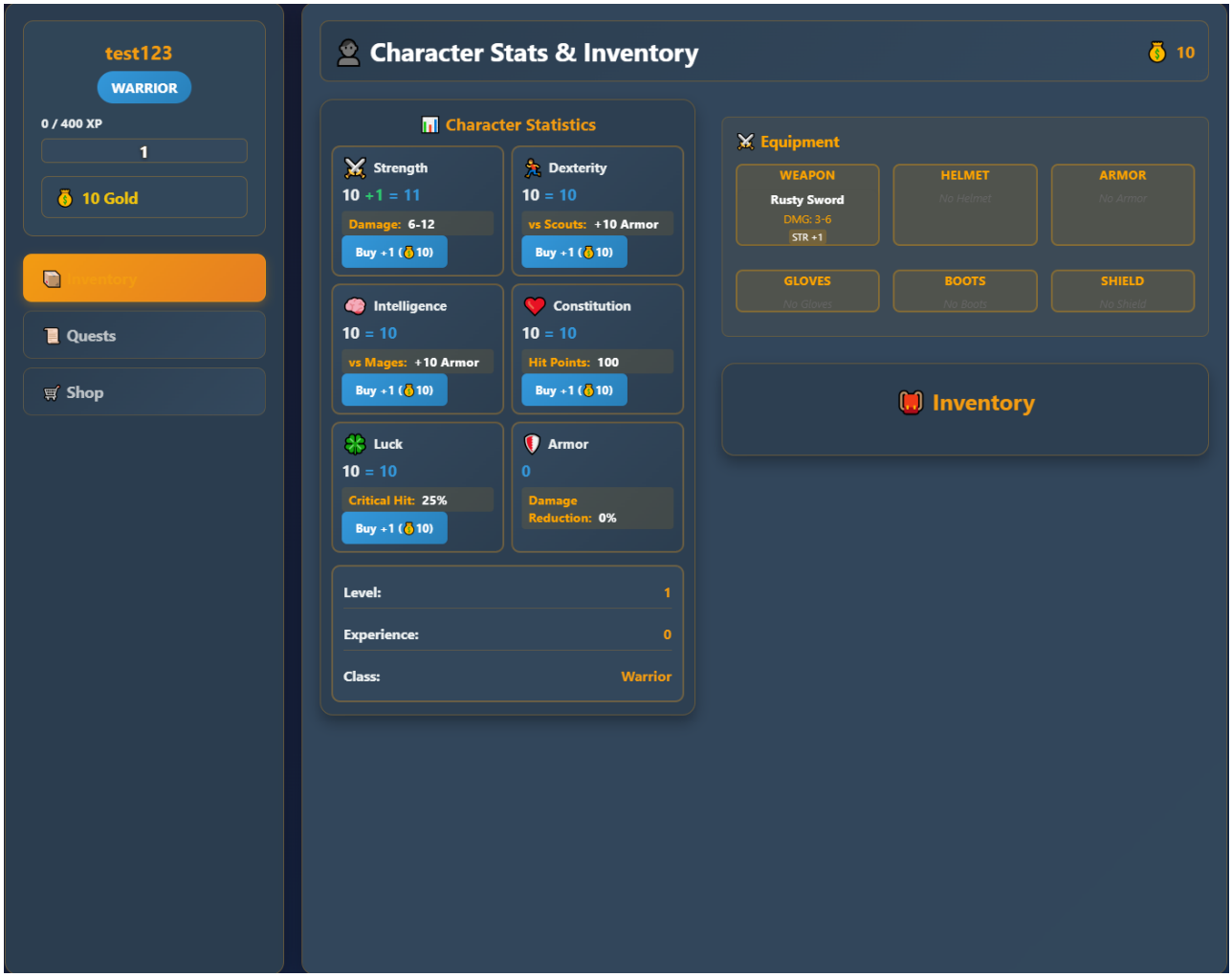
  – **Oczekiwany rezultat:** Przekierowanie do ekranu gry, widoczne dane gracza.

  – **Wynik testu:**



Rysunek 6.9: Test logowania użytkownika - frontend

- **Wyświetlanie ekwipunku gracza**
  - **Cel:** Sprawdzenie poprawności wyświetlania ekwipunku po zalogowaniu.
  - **Warunki początkowe:** Gracz jest zalogowany, posiada przedmioty.
  - **Kroki testowe:**
    1. Przejdź do ekranu ekwipunku.
  - **Dane wejściowe:** Token JWT (w tle).
  - **Oczekiwany rezultat:** Lista przedmiotów widoczna na ekranie.
  - **Wynik testu:**



Rysunek 6.10: Test ekwipunku gracza

- **Zakup przedmiotu przez interfejs**

  – **Cel:** Sprawdzenie możliwości zakupu przedmiotu przez UI.

  – **Warunki początkowe:** Gracz jest zalogowany, ma wystarczającą ilość złota.

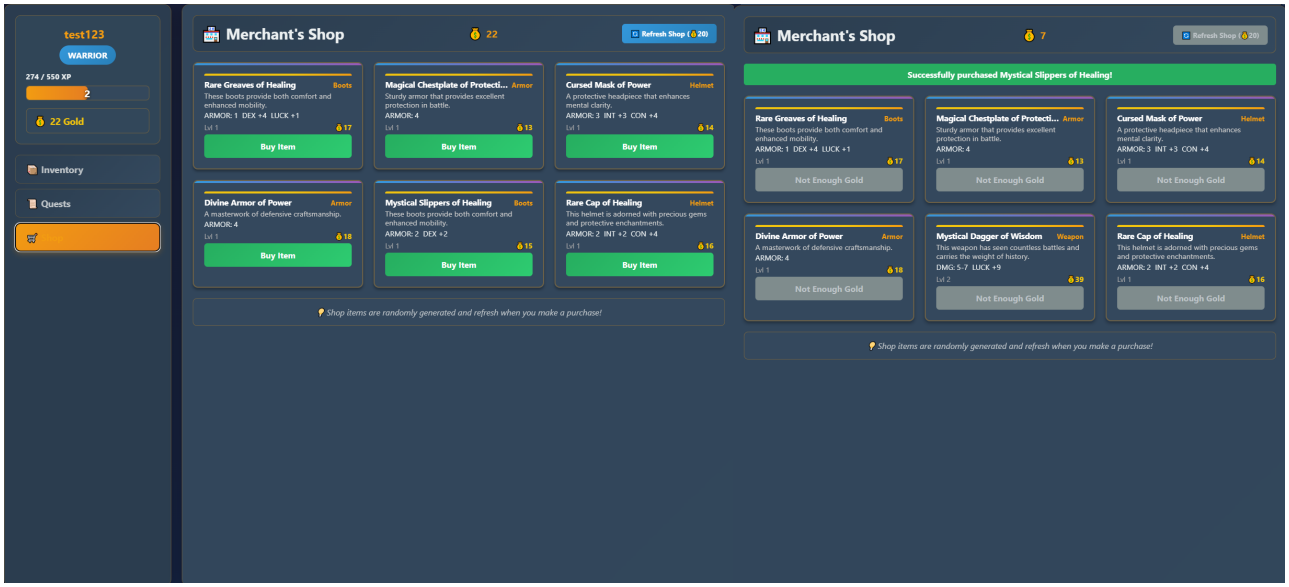
  – **Kroki testowe:**

    1. Przejdź do sklepu.
    2. Wybierz przedmiot i kliknij "Kup".

  – **Dane wejściowe:** Identyfikator przedmiotu (wybór w UI).

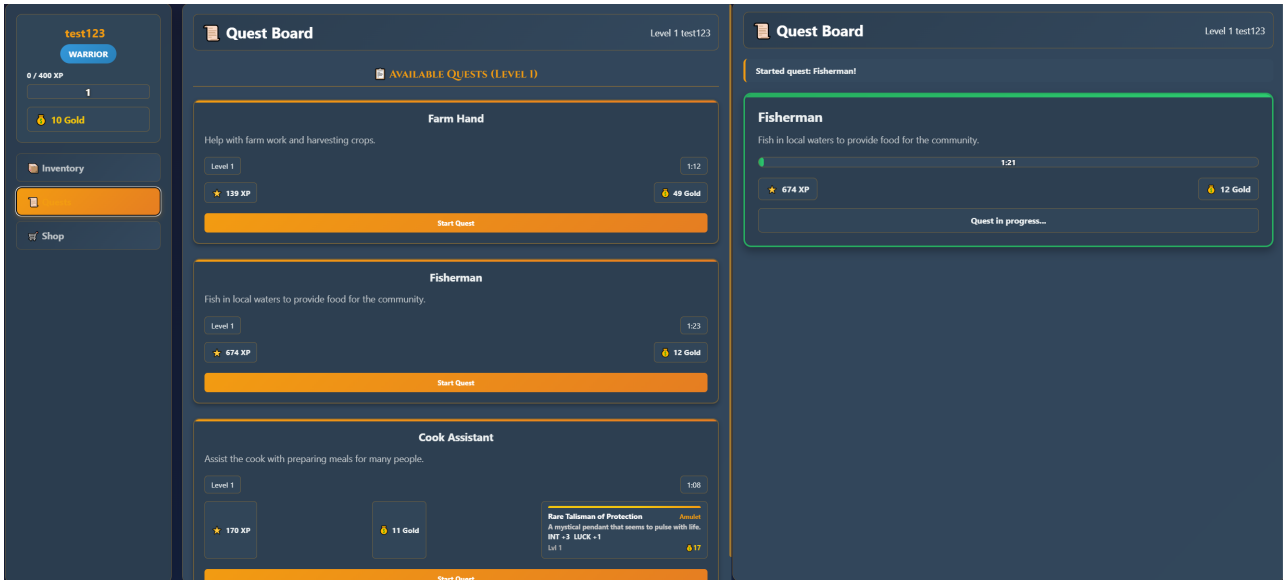
  – **Oczekiwany rezultat:** Przedmiot pojawia się w ekwipunku, złoto zmniejszone.

  – **Wynik testu:**

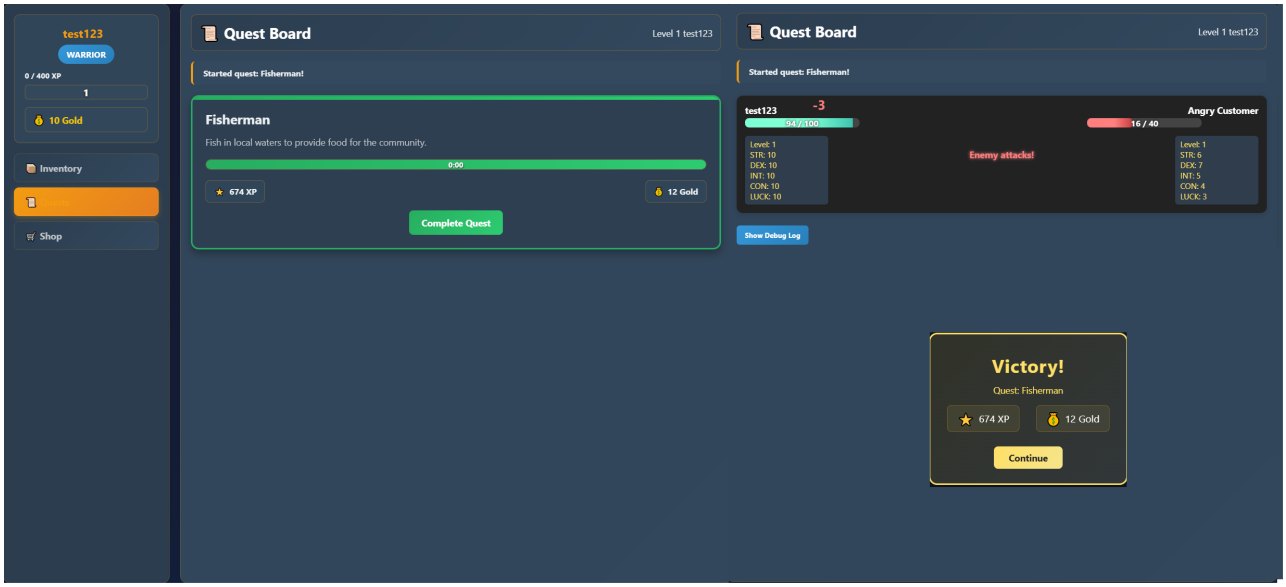


Rysunek 6.11: Test zakupu przedmiotu - frontend

- **Rozpoczęcie i zakończenie misji przez interfejs**
  - **Cel:** Sprawdzenie poprawności obsługi misji przez UI.
  - **Warunki początkowe:** Gracz jest zalogowany, ma dostępne misje.
  - **Kroki testowe:**
    1. Przejdź do tablicy misji.
    2. Wybierz misję i kliknij "Rozpocznij".
    3. Po zakończeniu kliknij "Odbierz nagrodę".
  - **Dane wejściowe:** Identyfikator misji (wybór w UI).
  - **Oczekiwany rezultat:** Misja znika z listy aktywnych, nagroda pojawia się w ekwipunku.
  - **Wynik testu:**

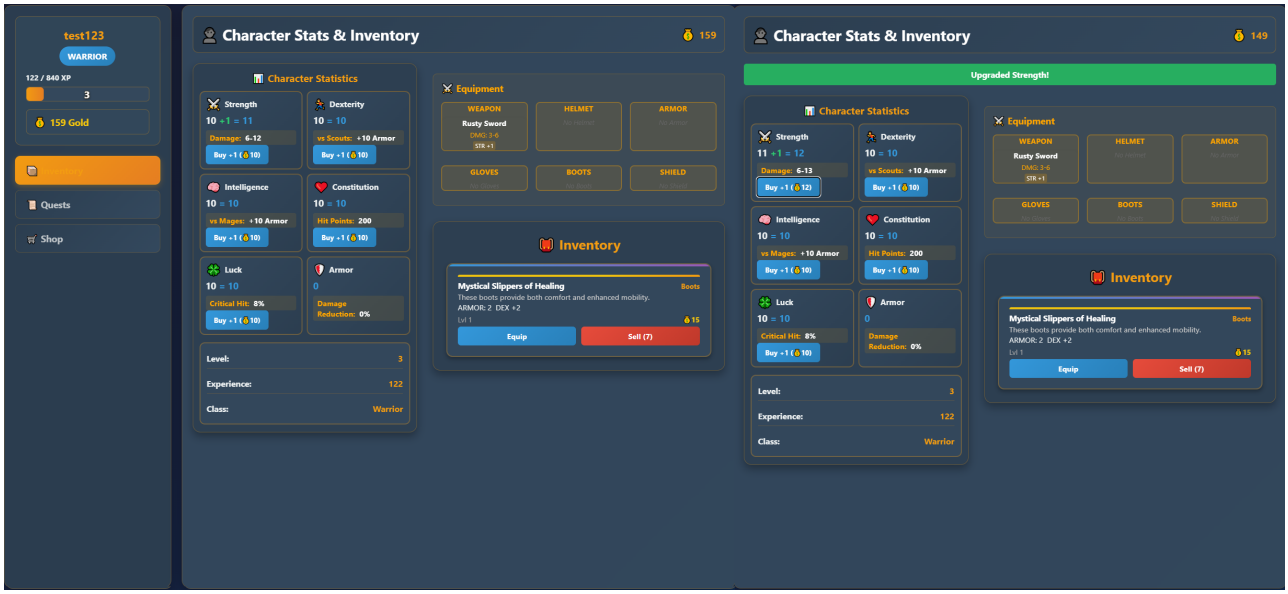


Rysunek 6.12: Test rozpoczęcia misji



Rysunek 6.13: Test zakończenia misji

- **Ulepszenie statystyk przez interfejs**
  - **Cel:** Sprawdzenie możliwości ulepszania statystyk przez UI.
  - **Warunki początkowe:** Gracz jest zalogowany, ma wystarczającą ilość złota.
  - **Kroki testowe:**
    1. Przejdź do ekranu statystyk.
    2. Kliknij przycisk ulepszenia wybranej statystyki.
  - **Dane wejściowe:** Nazwa statystyki (wybór w UI).
  - **Oczekiwany rezultat:** Statystyka zwiększona, złoto zmniejszone.
  - **Wynik testu:**



Rysunek 6.14: Test ulepszenia statystyk

Rozdział 7

Planowane place rozwojowe

W tej sekcji opisano kierunki rozwoju systemu, które mogą być zaimplementowane w przyszłych wersjach aplikacji.

7.1 Rozszerzenia funkcjonalności gry

- **System klanów i gildii**
  - Tworzenie i zarządzanie klanami przez graczy
  - Wspólne misje klanowe z większymi nagrodami
  - System rankingowy klanów
  - Czat wewnętrzny klanu
- **System PvP (Player vs Player)**
  - Areny PvP z rankingiem graczy
  - Turnieje sezonowe z nagrodami
  - System wyzwań między graczami
  - Specjalne przedmioty dostępne tylko w PvP
- **Rozszerzony system przedmiotów**
  - System rzadkości przedmiotów (pospolite, rzadkie, epickie, legendarne)
  - System enchantów i run
  - Kolekcjonowanie zestawów przedmiotów z bonusami
  - Skalowanie statystyk z poziomem
- **System osiągnięć**
  - Osiągnięcia za różne działania w grze
  - System punktów prestiżu
  - Specjalne tytuły i odznaki
  - Nagrody za osiągnięcia
- **Rozszerzenia fabularne**
  - Dodatkowe misje i przeciwnicy
  - System trudnych, nagradzających walk w lochach
  - Misje główne gry

7.2 Rozszerzenia techniczne

- **System powiadomień**
  - Powiadomienia push o zakończeniu misji
  - Powiadomienia e-mail o ważnych wydarzeniach
  - System przypomnień o aktywności
- **Optymalizacja wydajności**
  - Implementacja cache'owania na poziomie API
  - Optymalizacja zapytań do bazy danych
  - Lazy loading komponentów frontendu
  - Kompresja odpowiedzi API
- **Rozszerzenia bezpieczeństwa**
  - Rate limiting dla API
  - System wykrywania oszustw
  - Dwuetapowa weryfikacja (2FA)
  - Audit log wszystkich działań graczy
- **System backupów i recovery**
  - Automatyczne backupy bazy danych
  - System przywracania danych graczy
  - Replikacja bazy danych
  - Disaster recovery plan

7.3 Rozszerzenia interfejsu użytkownika

- **Responsywny design**
  - Pełna obsługa urządzeń mobilnych
  - Adaptacyjny layout dla różnych rozdzielczości
  - Touch-friendly interfejs
  - Progressive Web App (PWA)
- **Rozszerzone wizualizacje**
  - Animowane walki w czasie rzeczywistym
  - Efekty wizualne dla przedmiotów
  - System cząsteczek dla efektów specjalnych
  - Podstawowe grafiki, np. portrety graczy i przeciwników

7.4 Rozszerzenia społecznościowe

- **System czatu**
  - Czat globalny z moderacją
  - Czat prywatny między graczami
  - System emotikonów i reakcji
  - Filtrowanie treści
- **System przyjaciół**
  - Dodawanie przyjaciół
  - Lista online/offline
  - Wspólne misje z przyjaciółmi
  - System rekomendacji
- **System handlu**
  - Marketplace między graczami
  - System aukcji
  - Bezpieczne transakcje
  - Historia transakcji
- **System eventów**
  - Sezonowe eventy z nagrodami
  - Eventy weekendowe
  - System wyzwań czasowych
  - Specjalne misje eventowe

Bibliografia

[1] Dokumentacja .NET: https://docs.microsoft.com/pl-pl/dotnet/

[2] Dokumentacja ASP.NET Core: https://learn.microsoft.com/pl-pl/aspnet/core/

[3] Dokumentacja Angular: https://angular.io/docs

[4] Dokumentacja PostgreSQL: https://www.postgresql.org/docs/

[5] Dokumentacja Entity Framework Core: https://learn.microsoft.com/pl-pl/ef/core/

[6] Oficjalna dokumentacja JWT: https://jwt.io/introduction

[7] Własne doświadczenia i materiały dydaktyczne

Spis rysunków

Spis tabel