

Backup – „Dokumentacja”

Autor: Gniewomir Bartkowiak

Link do programu: <https://github.com/Gniewo1/Backup>

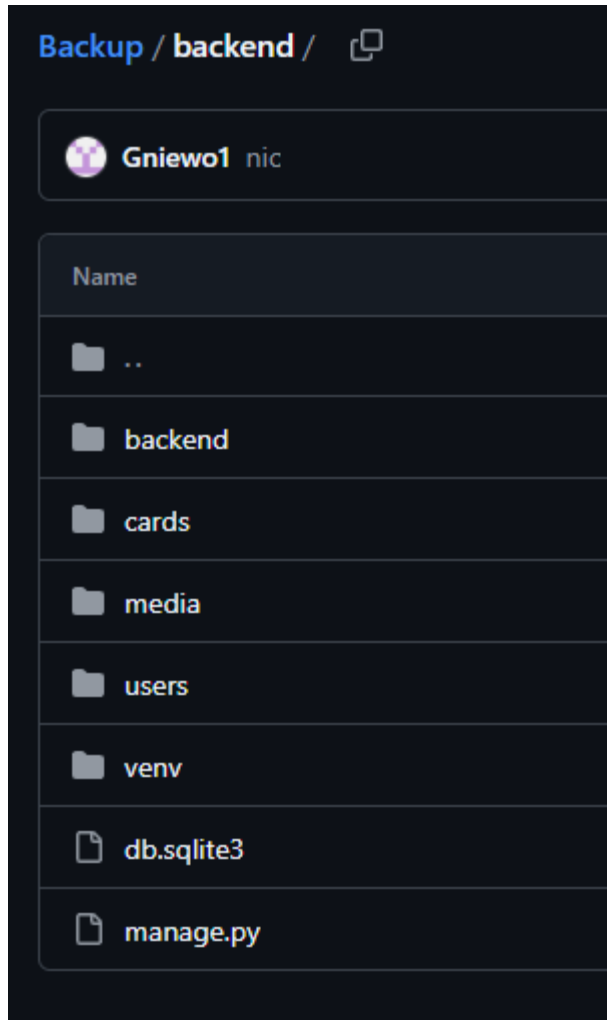
Wstęp:

Backup jest projektem strony internetowej w ramach mojej pracy inżynierskiej pt. „*System aukcyjny na przykładzie kolekcjonerskiej gry karcianej Magic the Gathering*”. Jest to marketplace, który ma umożliwiać kupno/sprzedaż karty do gry karcianej Magic the Gathering. Założenia projektu obejmowały:

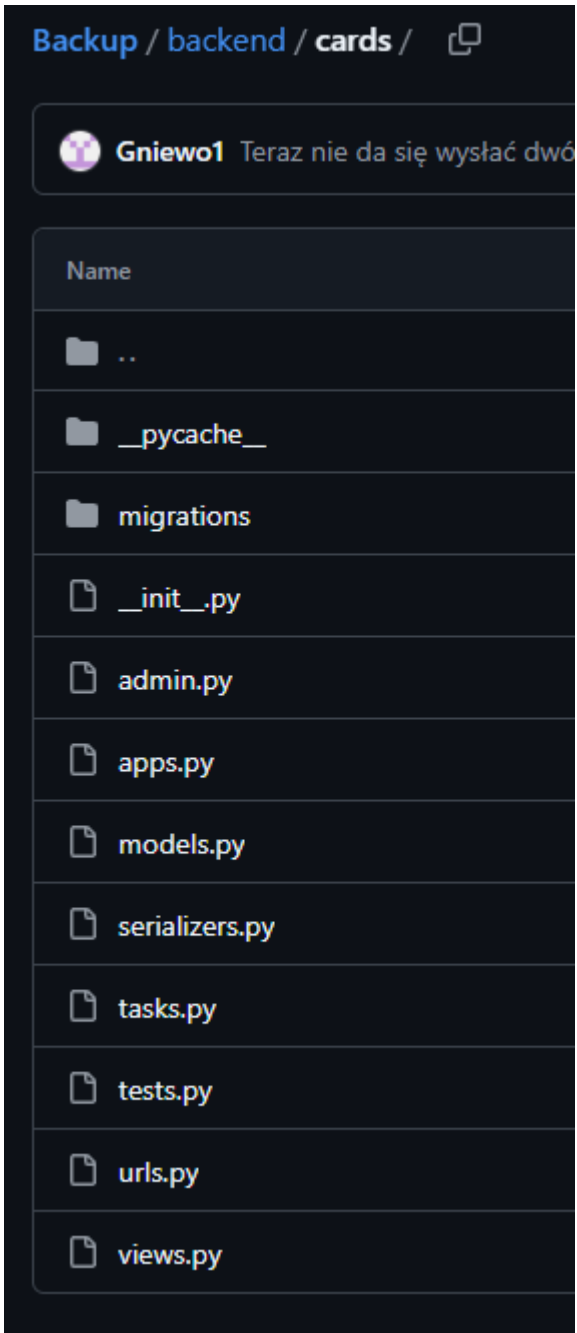
- System rejestracji oraz logowania
- Wystawianie na sprzedaż oraz kupno kart (możliwość stworzenia licytacji oraz ofert „kup teraz”)
- Wyszukiwanie ofert wraz z prostymi filtrami
- Przeglądanie ofert
- Finalizacja transakcji

Strona powstała przy użyciu **Django Rest Framework**(backend), biblioteki **React**(frontend) oraz **SQLite**(baza danych). Jeżeli się nie mylę (nie testowałem tego), żeby aplikacja działała należy zainstalować Pythona, Django oraz Reacta. Strona korzysta z innych bibliotek, ale jeśli chodzi o frontend są one w plikach projektu. Backend należy przed uruchomieniem serwera włączyć virtual environment, inaczej będą wyskakiwać błędy o braku zainstalowanej biblioteki.

Struktura plików:



Backend zawiera foldery widoczne na zdjęciu. Foldery **users** oraz **cards** są to dwie tzw. aplikacje stworzone w backendzie. Zawierają m.in. klasy, funkcje oraz adresy URL związane odpowiednio z użytkownikami(users) oraz kartami/ofertami kart(cards). Folder media zawiera wszystkie zdjęcia pojawiające się w projekcie. Folder **Backend** zawiera najważniejsze konfiguracje całego backendu. **Venv** zawiera pliki do włączenia virtual environment. Plik **db.sqlite3** bazą danych projektu, zaś manage.py jest integralną częścią Django i odpowiada za komendy.



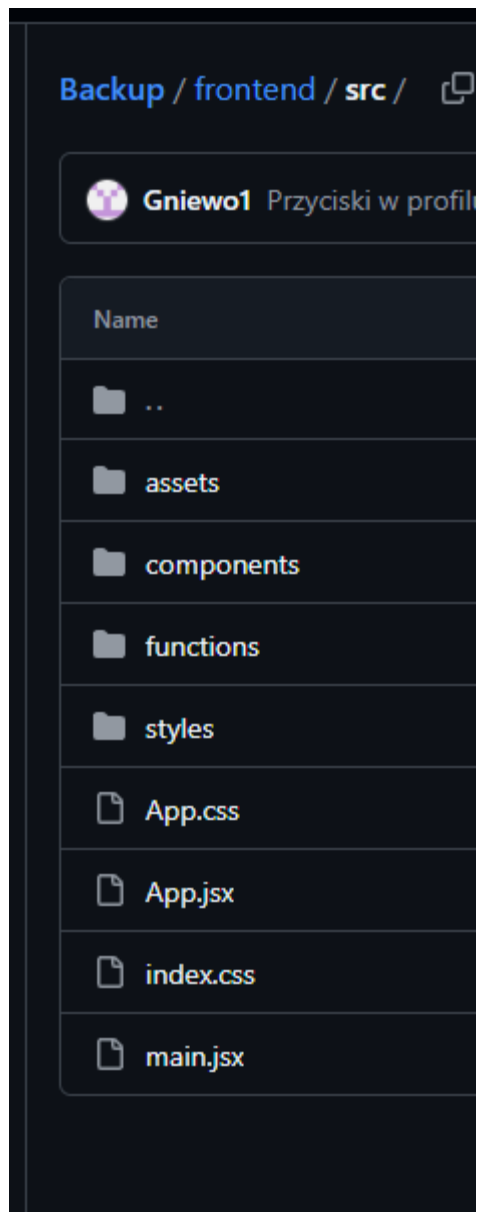
W przykładowej aplikacji, w tym wypadku cards. Opiszę tylko najważniejsze.

Models.py zawiera klasy tworzące tabele w bazie danych.

Views.py funkcje (czasami klasy, które posiadają funkcje) pozwalające na wprowadzanie/pobieranie/modyfikowanie danych w bazie danych.

Urls.py tworzy adresy API do wcześniej stworzonych funkcji. API to później są wykorzystywane we frontendzie.

Serializers.py zawiera klasy, które mogą być używane przez funkcje w view.py. Serializers zmienia format danych Django na JSON. Serializers nie są niezbędne. W projekcie w ramach czasami ich używałem, a czasami nie. Szczerze sam nie wiem, co jest lepszą praktyką. Używanie ich czy nie.



We Frontendzie pokazuje tylko src, ponieważ reszta plików powstały po instalacji i nic tam nie zmieniałem.

Components zawiera komponenty wykorzystywane na stronie. Sam React opiera się na tym, że tworzy się komponenty, które później używa się wielokrotnie w różnych stronach. W przypadku mojego projektu, każdy komponent jest osobną stroną, za wyjątkiem Navbar-u, który jest dodawany do każdej strony.

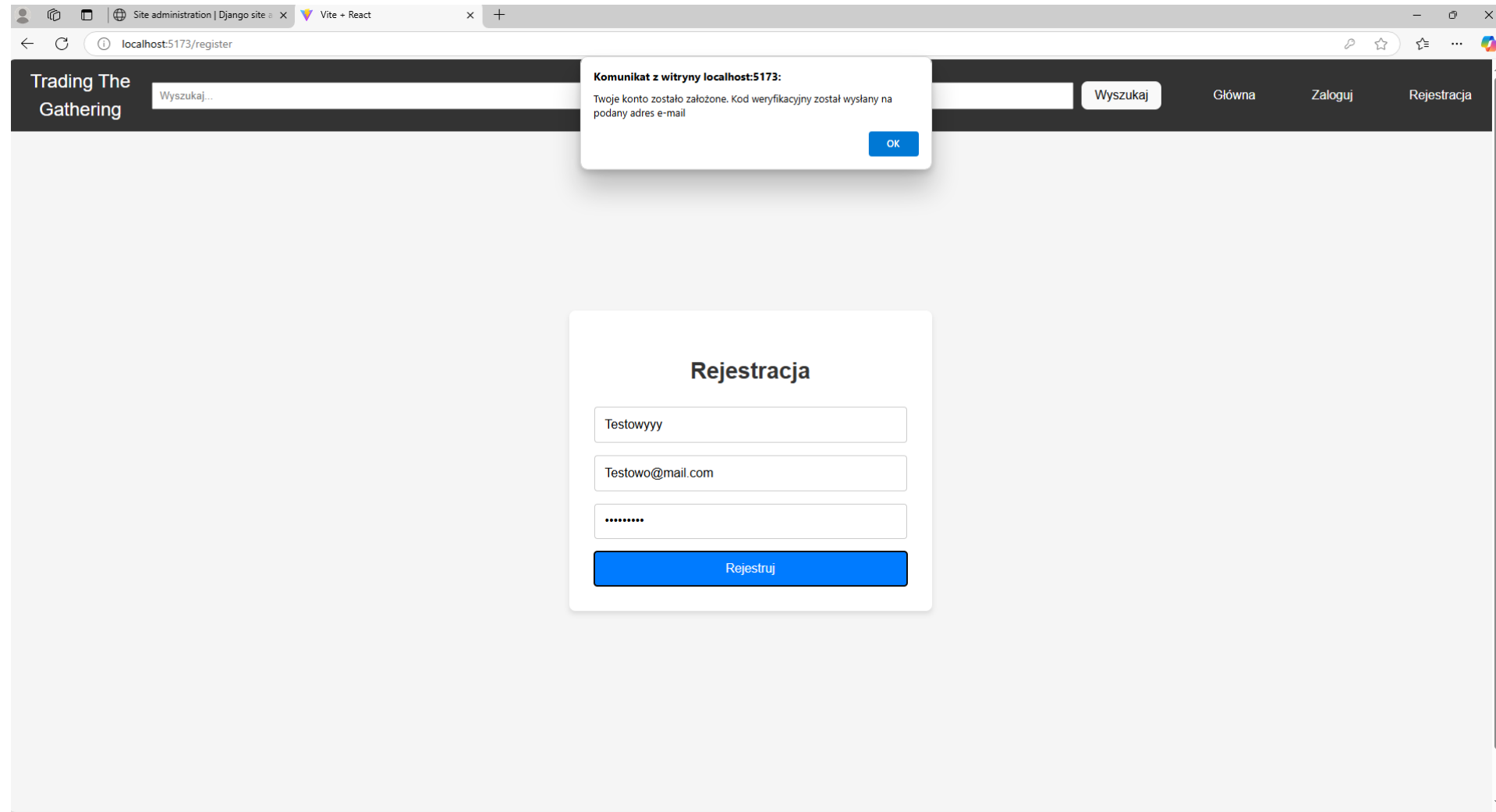
Functions zawiera funkcję, które jak na początku myślałem będę często używał. W ramach prac okazało się, że żadnej z nich nie używałem więcej niż raz, więc w teorii ten folder jest niepotrzebny.

Styles zawiera pliki CSS. Z racji na raczej chaotyczny proces tworzenia frontendu ten folder to jeden wielki bałagan. Każdy plik CSS w tym folderze odnosi się do innej strony, co nie powinno mieć miejsca. Jakbym drugi raz się za to zabierał, pliki odnosiłyby się bardziej do określonych elementów np. buttons, labels, a nie do konkretnej strony.

App.jsx zawiera strukturę adresów URL we frontendzie

Logowanie/Rejestracja

Rejestracja oprócz tego, że tworzy konto, połączone jest G-mailem i wysyła kod weryfikacyjny na podany adres e-mail. Logowanie jest tokenowe i korzysta z pakietu uwierzytelniania Knox.



[Register.jsx](#)

Żeby móc wejść na profil i móc kupować/sprzedawać karty należy najpierw zweryfikować konto

Trading The Gathering

Wyszukaj...

Wyszukaj

Główna

Profil

Wyloguj

Podaj kod weryfikacyjny

673283

Weryfikuj

Verification successful

[Verification.jsx](#)

Wystawianie kart – żeby wystawić kartę trzeba najpierw ją znaleźć. Strona przed załadowaniem pobiera wszystkie nazwy kart będące w bazie danych. Dzięki temu może po wpisywaniu podpowiadać nazwy, co widać na 1. screenie. Po wyborze karty strona pokazuje ją co widać na 2. screenie. Istnieje możliwość ustawienia typu oferty. Jako, że strona nie zakładała czatu, wymagane jest podanie nr konta już przy tworzeniu oferty

Sprzedaj kartę

Fireball
Forest
Wrath of God

Typ oferty:
Kup Teraz oraz Aukcja

Cena Kup Teraz [zł] :

Cena Aukcji [zł] :

Numer konta bankowego:

Przód karty:
Wybierz plik Nie wybrano pliku

Tył karty:
Wybierz plik Nie wybrano pliku

Sprzedaj kartę

Fireball

Fireball



Fireball

Sorcery

This spell costs 1 more to cast for each target beyond the first.
Fireball deals X damage divided evenly, rounded down, among any number of targets.
The orchard soon reeked of sulfur, smoke, and scorched bugbear, with just a hint of peach.

175/341 U
CLB • EN • XAVIER RIBEIRO

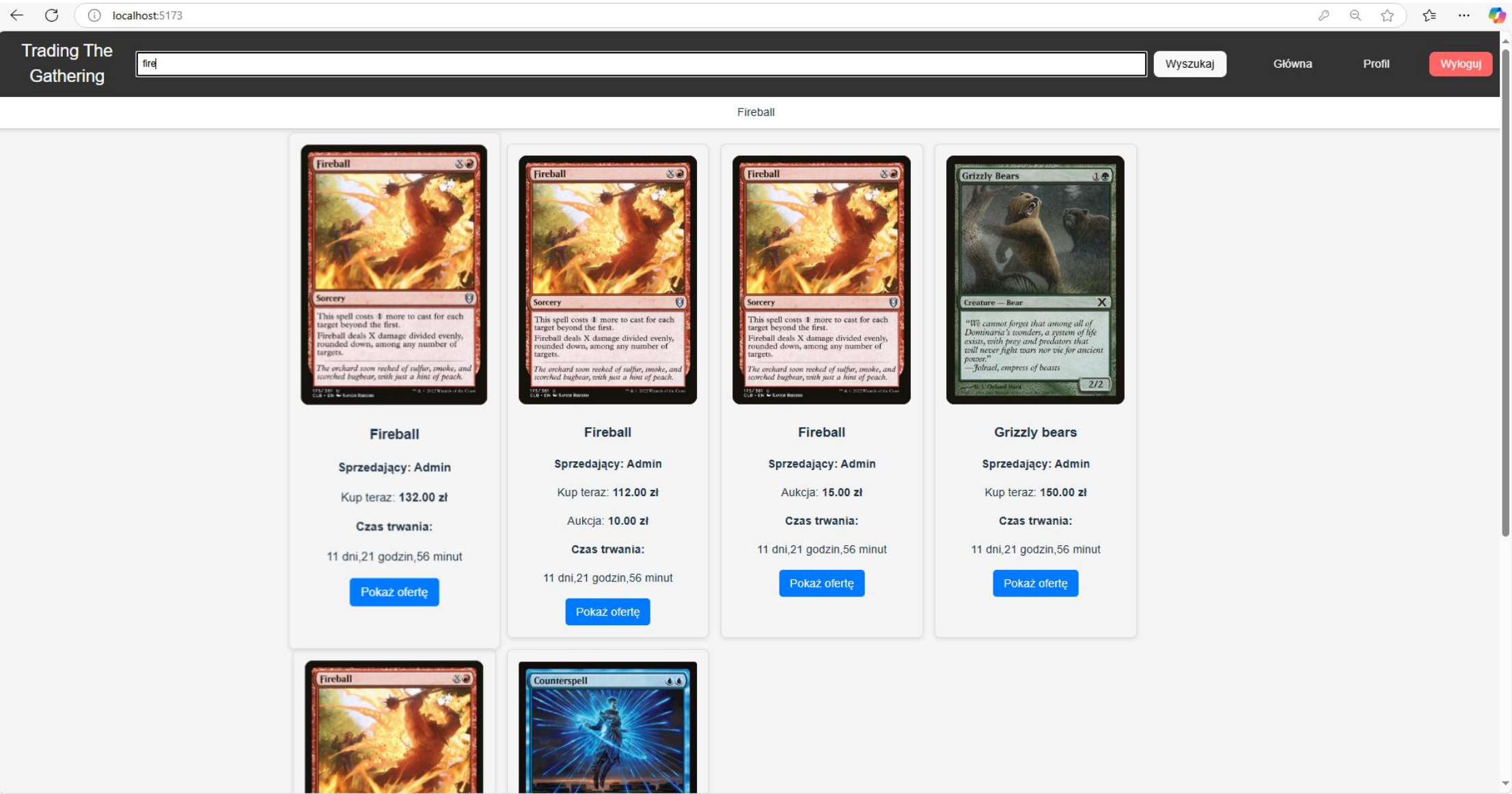
Typ oferty:
Aukcja

Cena Aukcji [zł] :

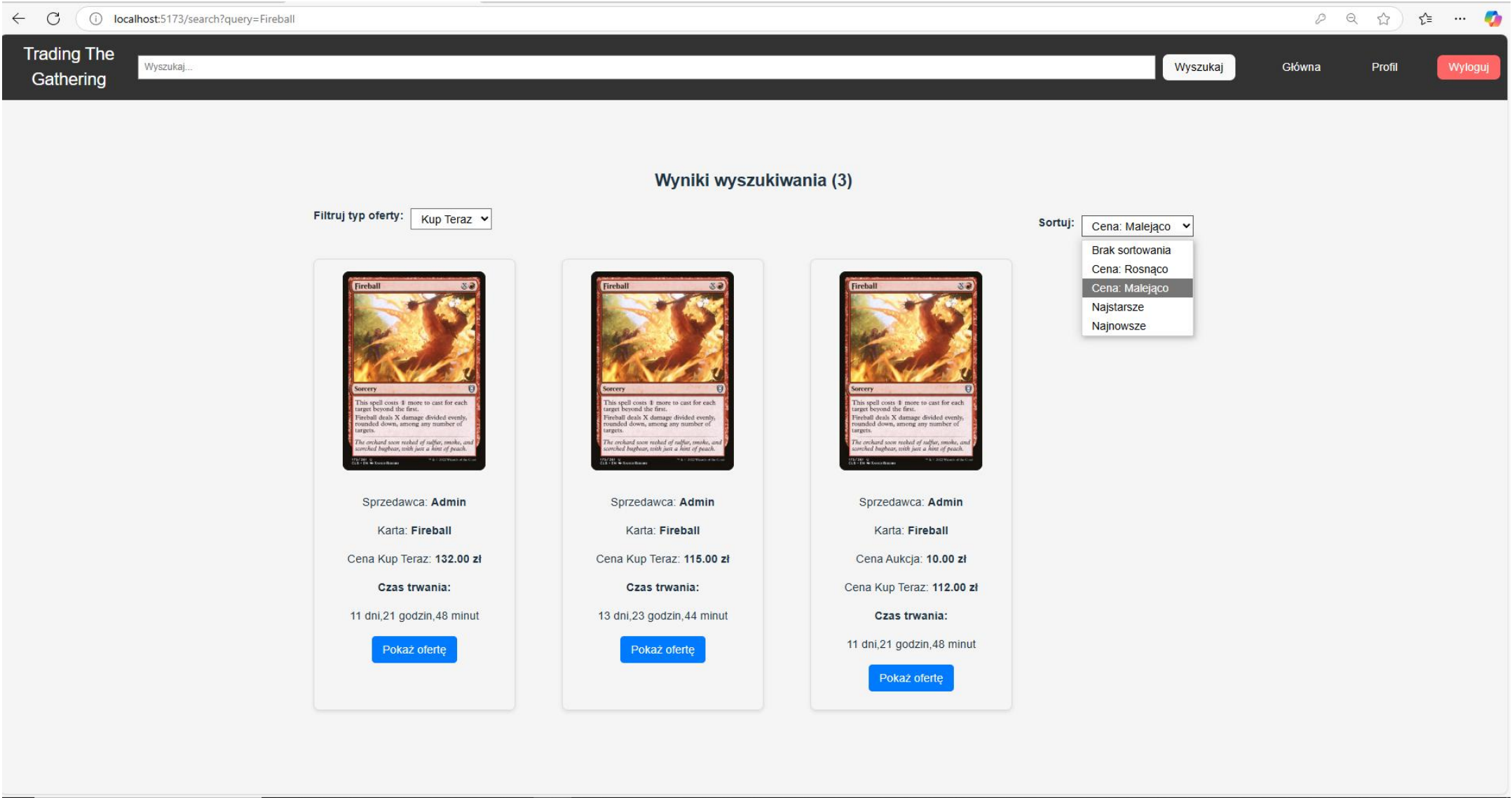
Numer konta bankowego:

[SellCard.jsx](#)

Strona startowa pokazuje do 15 najnowszych aktywnych ofert. Na górze tak jak na każdej innej stronie jest pasek nawigacyjny. Na nim jest **wyszukiwarka**, która tak samo jak przy tworzeniu oferty podpowiada nazwy kart



Wyszukiwarka posiada dwa filtry. Jeden filtruje typy oferty drugi sortuje w zależności od ceny albo od czasu trwania



[SearchResults.jsx](#)

Przegląd oferty, który jednocześnie umożliwia zakup albo zaliczycie kartę. Przycisk z dołu po lewej pozwala na pokazanie przodu lub tyłu karty.

Trading The Gathering

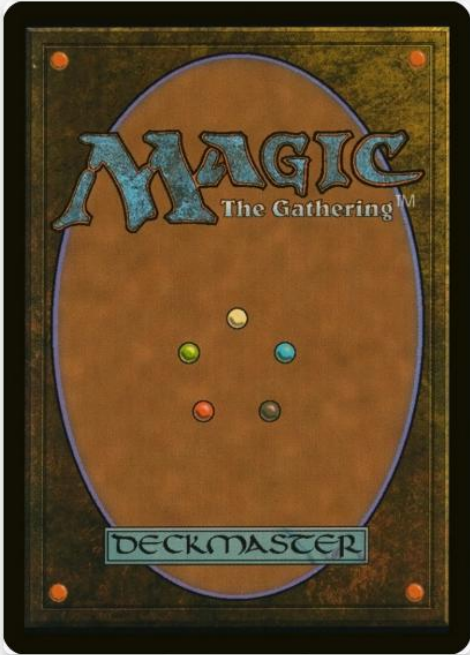
Wyszukaj

Główna

Profil

Wyloguj

Fireball



Pokaż Przód

Sprzedawca: Admin

Aukcja Cena: 10.00 zł

Dodaj swoją ofertę

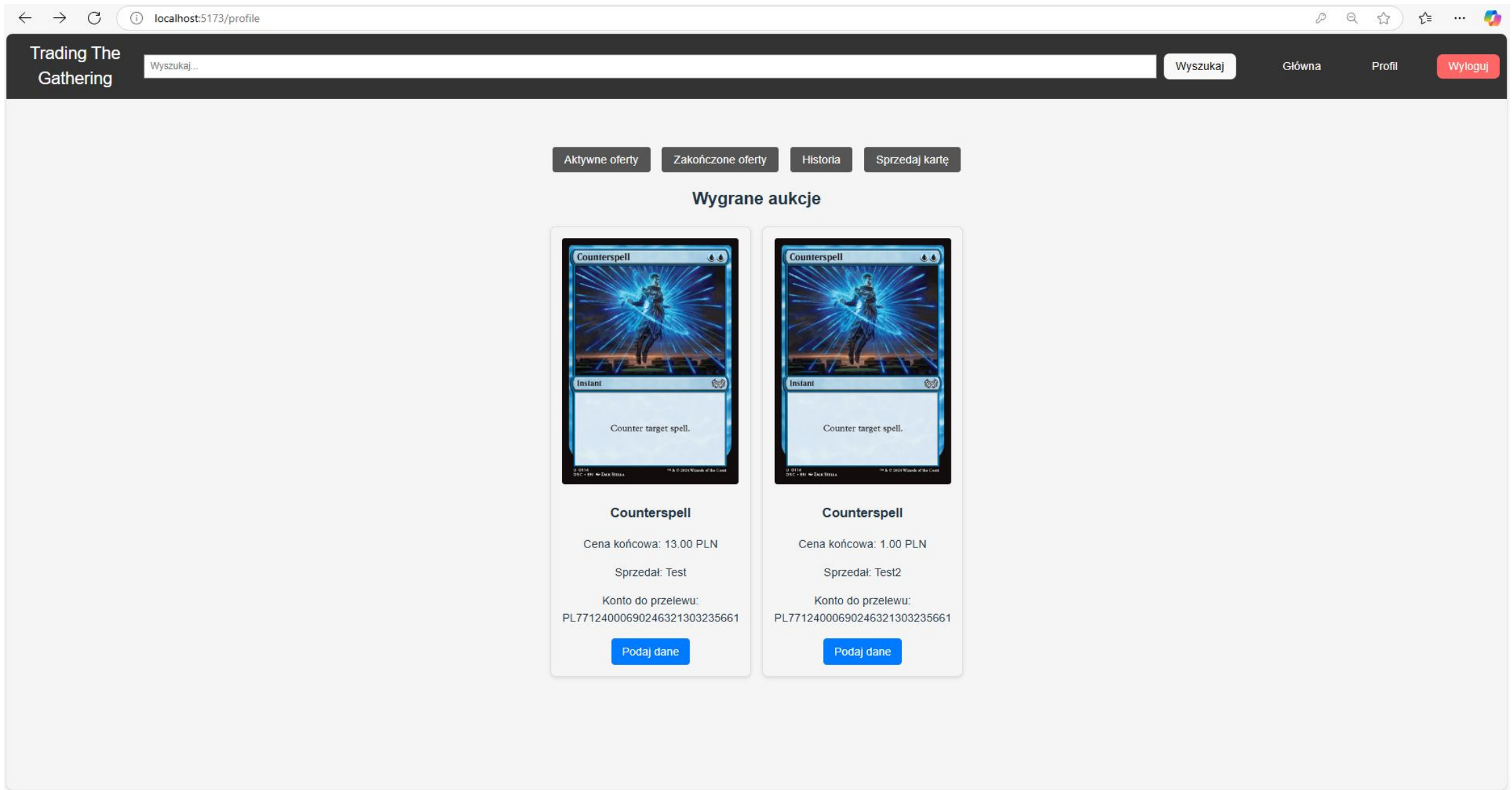
Kup Teraz Cena: 112.00 zł

Kup Teraz

Czas trwania:

11 dni, 21 godziny, 45 minut

Widok z profilu użytkownika. Z braku czasu niestety dosyć ubogi. Jedynie pokazuje wygrane oferty. Z czterech przycisków tylko „Sprzedaj kartę” przekierowuje do strony z podpunktu wystawianie kart



Widok gdzie po wygranej aukcji można dodać swoje dane do przesyłki

Trading The Gathering


Wyszukaj

Główna

Profil

Wyloguj

Counterspell



Counter target spell.

U 0114
DSC • EN • ZACK STELLA

™ & © 2024 Wizards of the Coast

Pokaż Tył

Sprzedawca: **Test**

Cena końcowa: 13.00 PLN

Konto do przelewu:
PL77124000690246321303235661

Wymagane dane do wysyłki

Imię

Nazwisko

Ulica

Numer budynku

Numer mieszkania

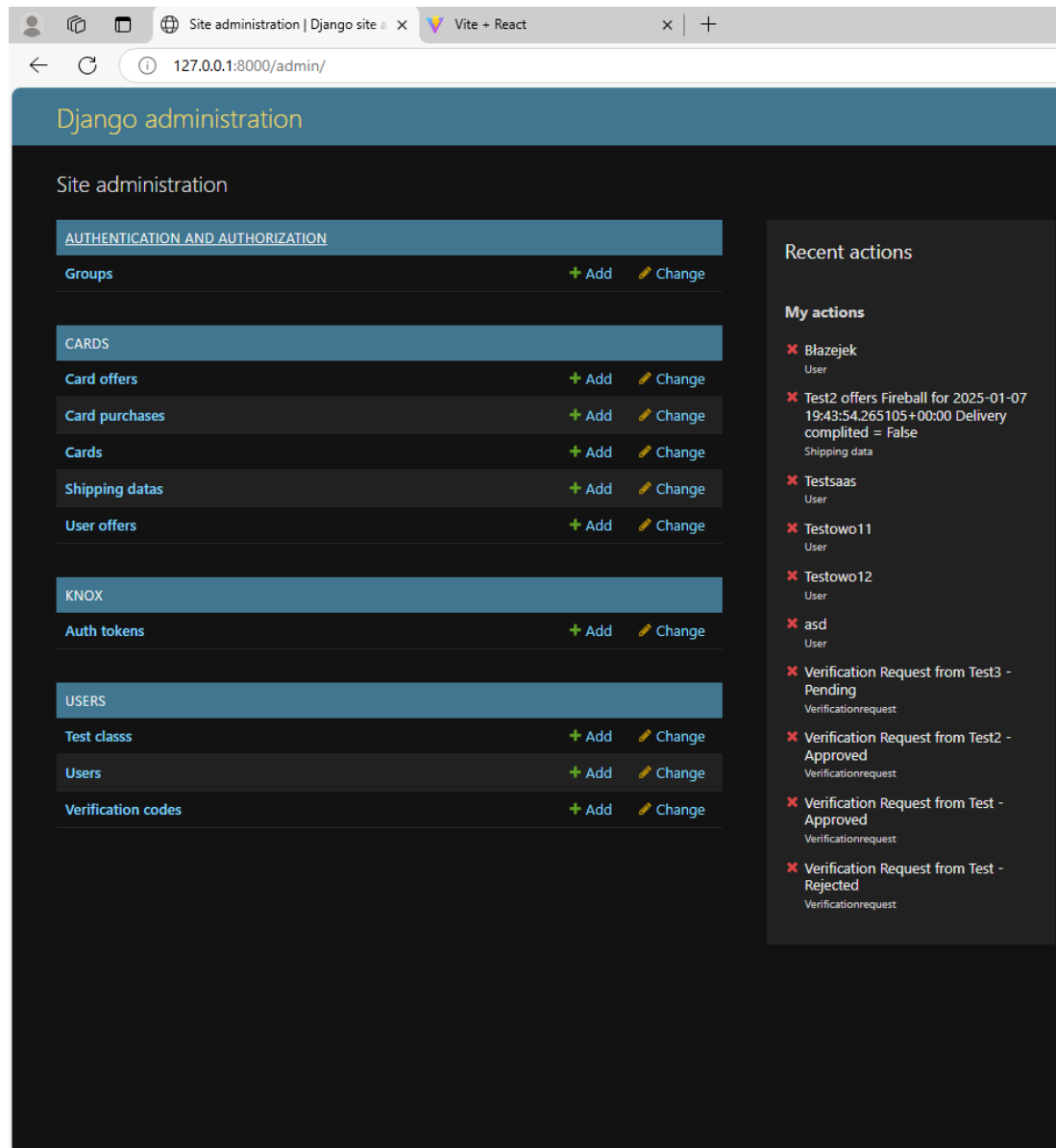
Miasto

Kod pocztowy

Zatwierdź dane do wysyłki

[Shipment.jsx](#)

Widok z panelu administratora na bazę danych



Card offers jest to tabela zawierająca oferty kart. W *User offers* rekordy tworzone są gdy jakiś użytkownik zaliczytuje, kupi jakąś kartę. Każdy *User offers* połączony jest z daną *Card offers*, podobnie jak *Shipping data* które przechowuje adresy. W *Cards* są zapisane karty (nazwa razem ze zdjęciem).

Card purchases oraz *Test classs* nie są wykorzystywane. Są pozostałością po testach.

Reszta raczej mówi sama za siebie. Jedynie mogę dodać, że klasa *Users* została zmodyfikowana.

Więcej informacji w plikach *models.py* w folderach *users* i *cards*.

Testy jednostkowe – pisane były na samym końcu. Jako, że robiłem i backend i frontend, testy przeprowadzałem w locie, od razu na działającej (lub nie) stronie. Testy jednostkowe wykonałem bardziej po to, żeby by i w ogóle zobaczyć jak to się robi. Są one tylko tu [tests.py](#)