

Backup – „Documentation”

Author: Gniewomir Bartkowiak

Link to program: <https://github.com/Gniewo1/Backup>

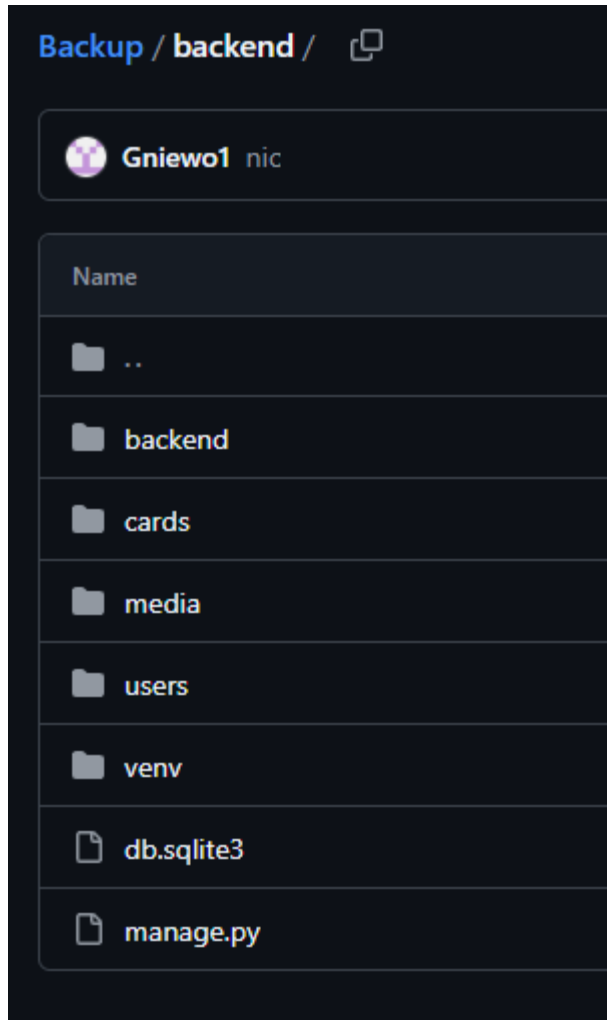
Entry:

Backup is a website project developed as part of my engineering thesis titled "*The auction system based on the example of the collectible card game Magic the Gathering*". It is a marketplace designed to enable the buying and selling of Magic: The Gathering trading cards. The project assumptions included:

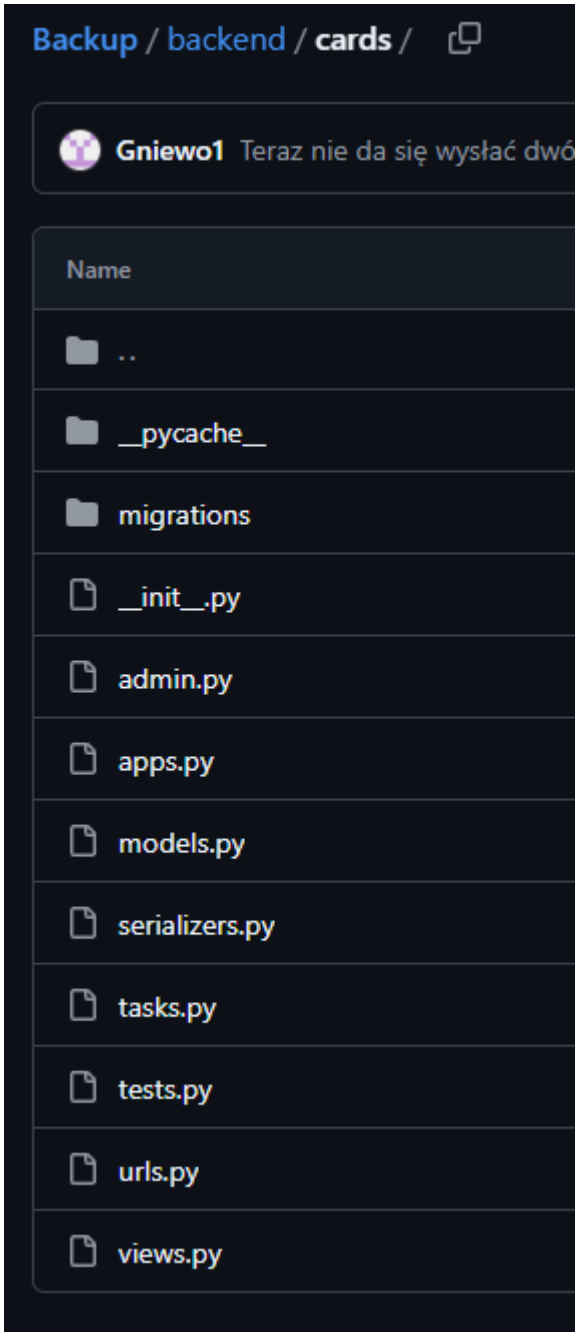
- Registration and login system
- Posting and buying cards (possibility to create auctions and "buy now" offers)
- Search offers with simple filters
- Browse offers
- Finalization of the transaction

The website was created using **Django Rest Framework** (backend), **React** (frontend), and **SQLite** (database). If I'm not mistaken (I haven't tested this), to make the application work, Python, Django, and React need to be installed. The website uses other libraries, but as far as the frontend is concerned, they are in the project files. The backend requires enabling a virtual environment before starting the server; otherwise, errors about missing libraries will occur.

File structure:



The backend contains folders visible in the picture. The **users** and **cards** folders are two so-called applications created in the backend. They contain classes, functions, and URLs related to users (users) and cards/card offers (cards). The **media** folder contains all the images used in the project. The **Backend** folder contains the most important configurations for the entire backend. The **venv** folder contains files to enable the virtual environment. The **db.sqlite3** file is the project's database, and **manage.py** is an integral part of Django, responsible for running commands.



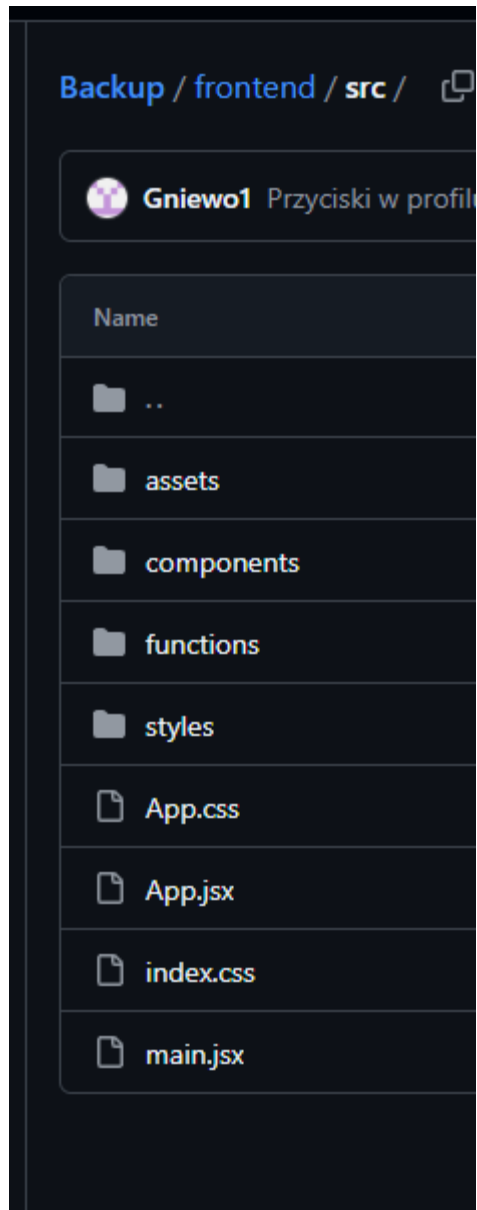
In the example application, in this case, **cards**, I will describe only the most important parts.

models.py contains classes that create tables in the database.

views.py contains functions (sometimes classes that have functions) that allow for adding/retrieving/modifying data in the database.

urls.py creates API endpoints for the previously created functions. These APIs are later used in the frontend.

serializers.py contains classes that can be used by functions in **views.py**. Serializers convert Django data into JSON format. Serializers are not essential. In the project, I sometimes used them, and sometimes I didn't. Honestly, I'm not sure what the best practice is—whether to use them or not.



In the frontend, only the **src** folder is shown because the rest of the files were created after installation, and I didn't modify them.

Components contains components used on the website. React is based on creating components that are later used multiple times across different pages. In my project, each component is a separate page, except for the **Navbar**, which is added to every page.

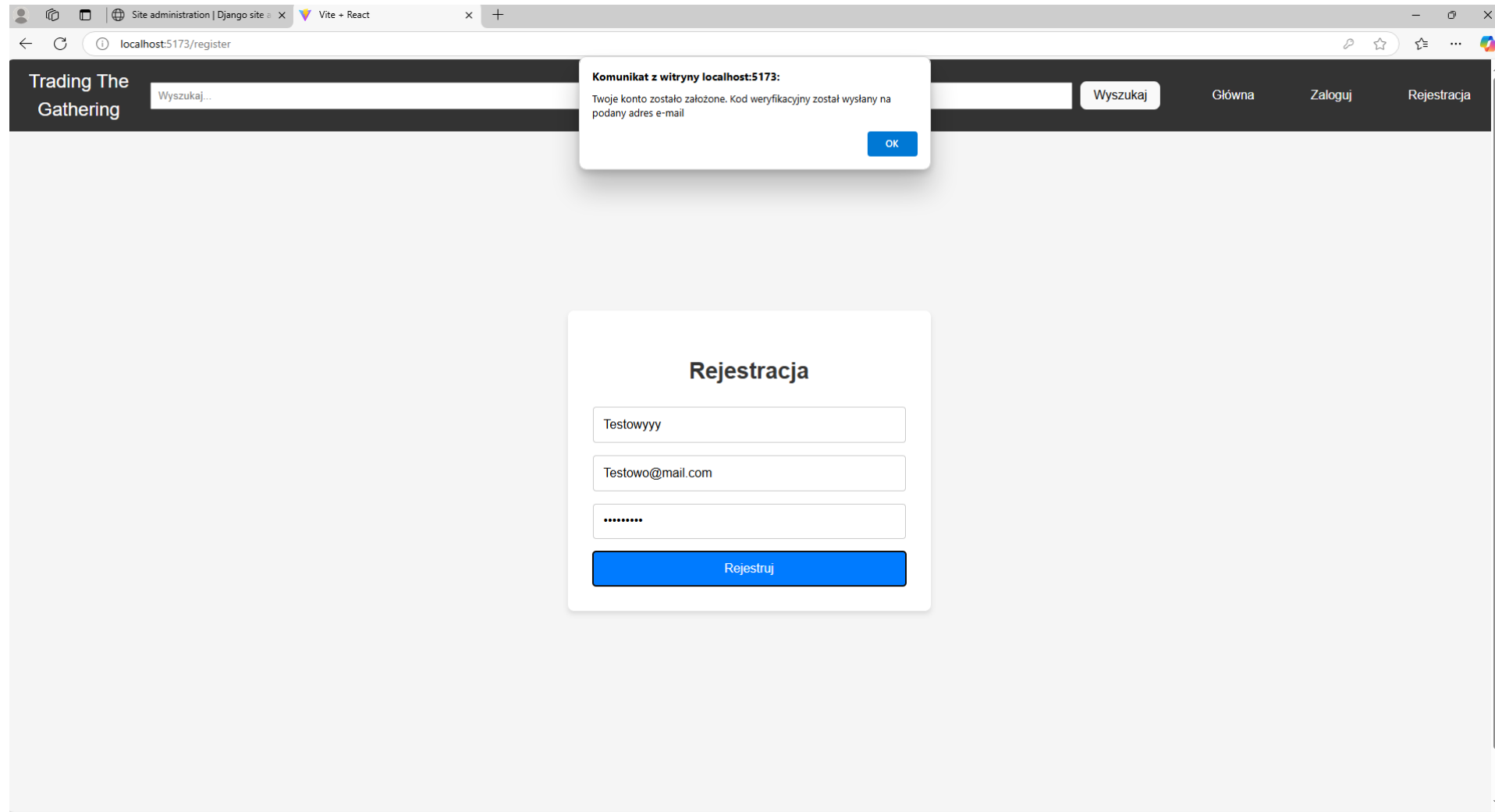
Functions contains functions that I thought I would use frequently at the beginning. However, as the work progressed, it turned out that none of them were used more than once, so in theory, this folder is unnecessary.

Styles contains CSS files. Due to the rather chaotic process of frontend development, this folder is a complete mess. Each CSS file in this folder corresponds to a different page, which shouldn't be the case. If I were to do it again, the files would be more focused on specific elements, such as buttons or labels, rather than being tied to a specific page.

App.jsx contains the URL structure for the frontend.

Login/Registration

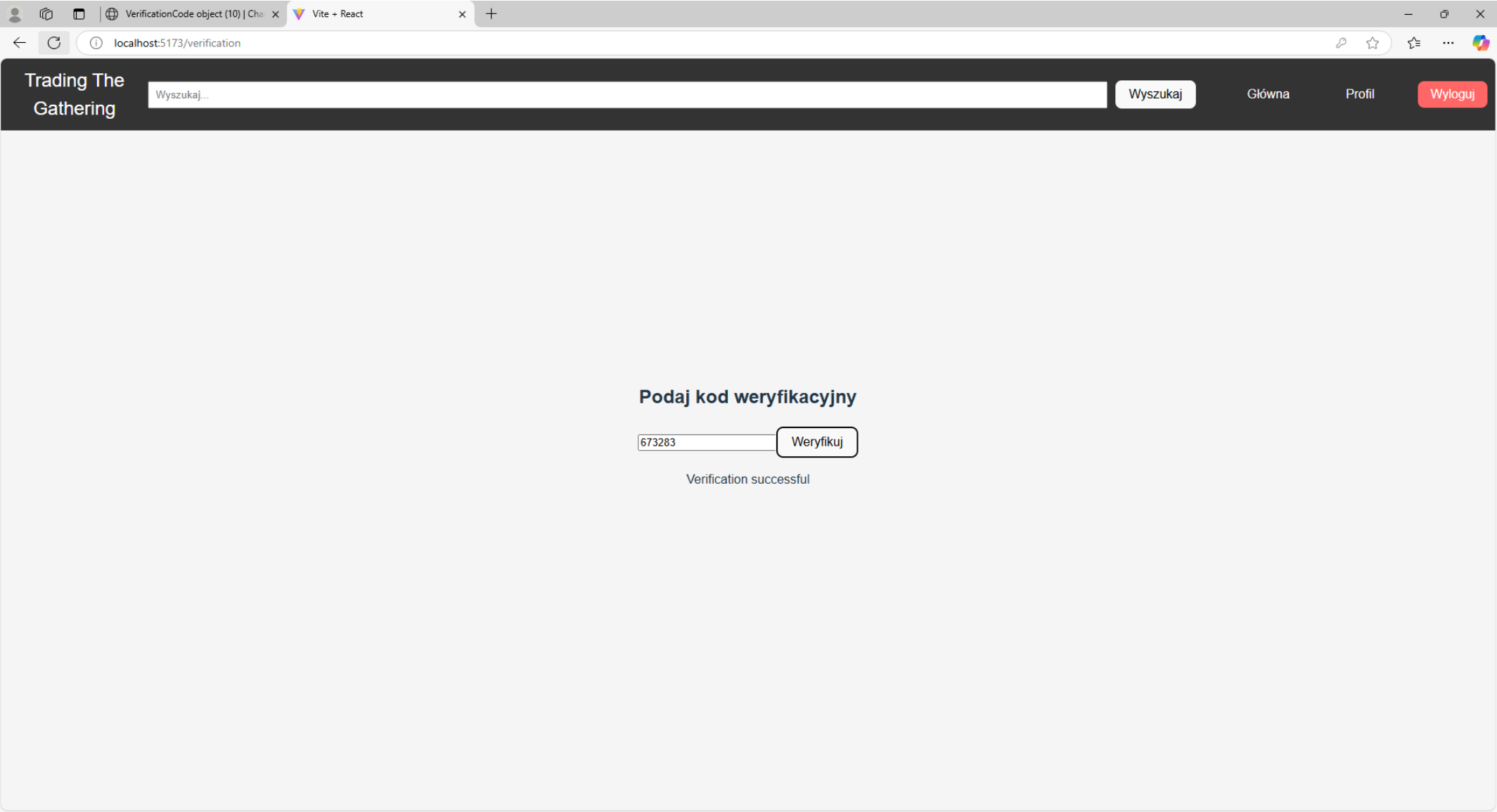
Registration system, in addition to creating an account, links it with Gmail and sends a verification code to the provided email address. Login system is token-based and uses the Knox authentication package.



The screenshot shows a web browser window with the URL `localhost:5173/register`. The browser tabs include "Site administration | Django site" and "Vite + React". The page header features the logo "Trading The Gathering" on the left, a search bar with the placeholder "Wyszukaj...", and navigation links "Główna", "Zaloguj", and "Rejestracja" on the right. A modal dialog box is displayed in the center of the screen, titled "Rejestracja". It contains three input fields: the first contains "Testowyyy", the second contains "Testowo@mail.com", and the third is a password field with masked characters "*****". Below these fields is a blue button labeled "Rejestruj". Above the registration form, another modal dialog box is visible, titled "Komunikat z witryny localhost:5173:", containing the text "Twoje konto zostało założone. Kod weryfikacyjny został wysłany na podany adres e-mail" and an "OK" button.

[Register.jsx](#)

In order to be able to enter your profile and buy/sell cards, you must first verify your account.



[Verification.jsx](#)

Listing cards for sale – to list a card, you first need to find it. The website retrieves all card names from the database before loading the page. This allows it to suggest names as you type, as seen in the first screenshot. After selecting a card, the website displays it, as shown in the second screenshot. There is an option to set the type of offer. Since the website does not include a chat feature, providing an account number is required when creating an offer.

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



Typ oferty:

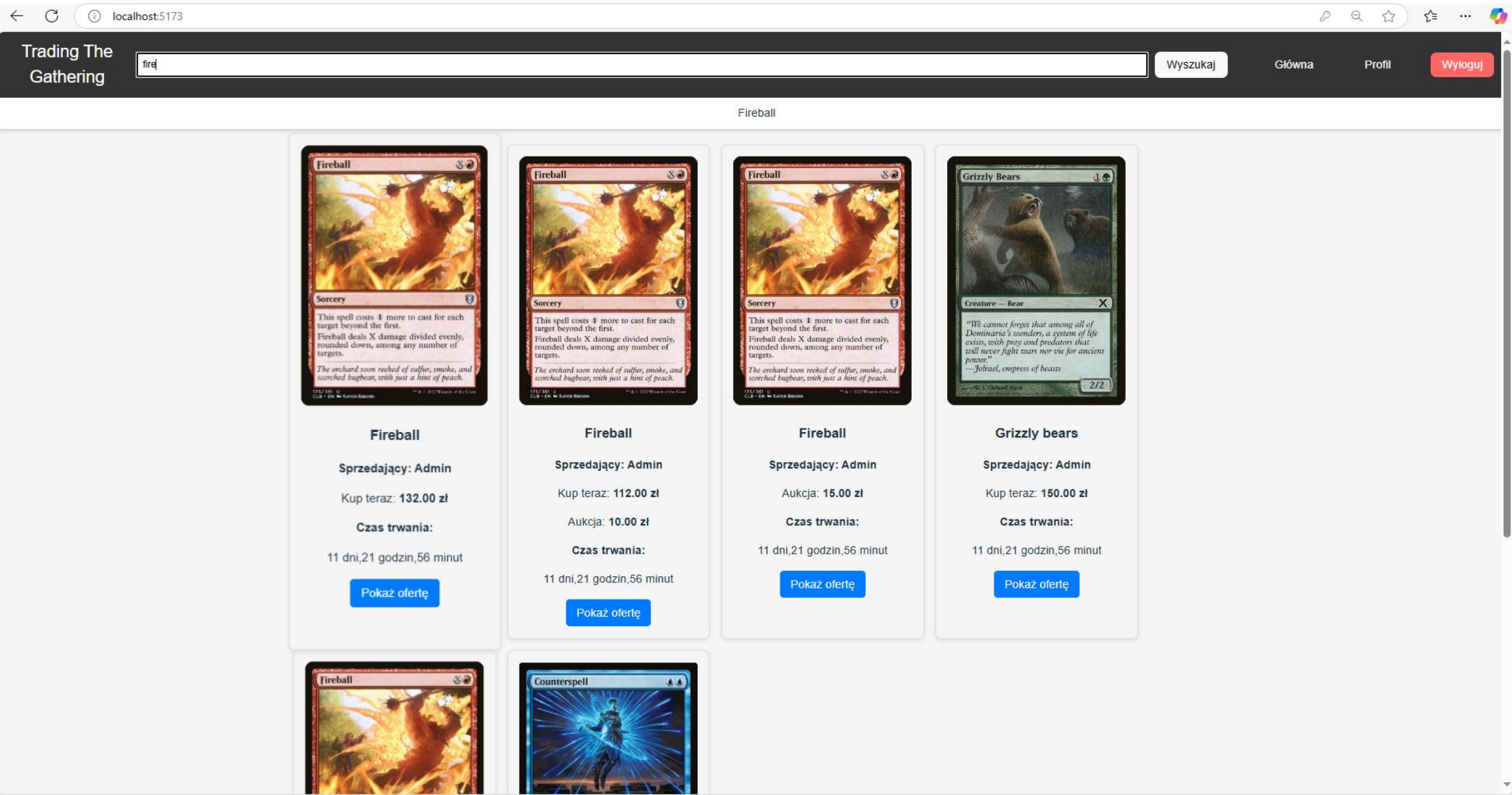
Aukcja

Cena Aukcji [zł] :

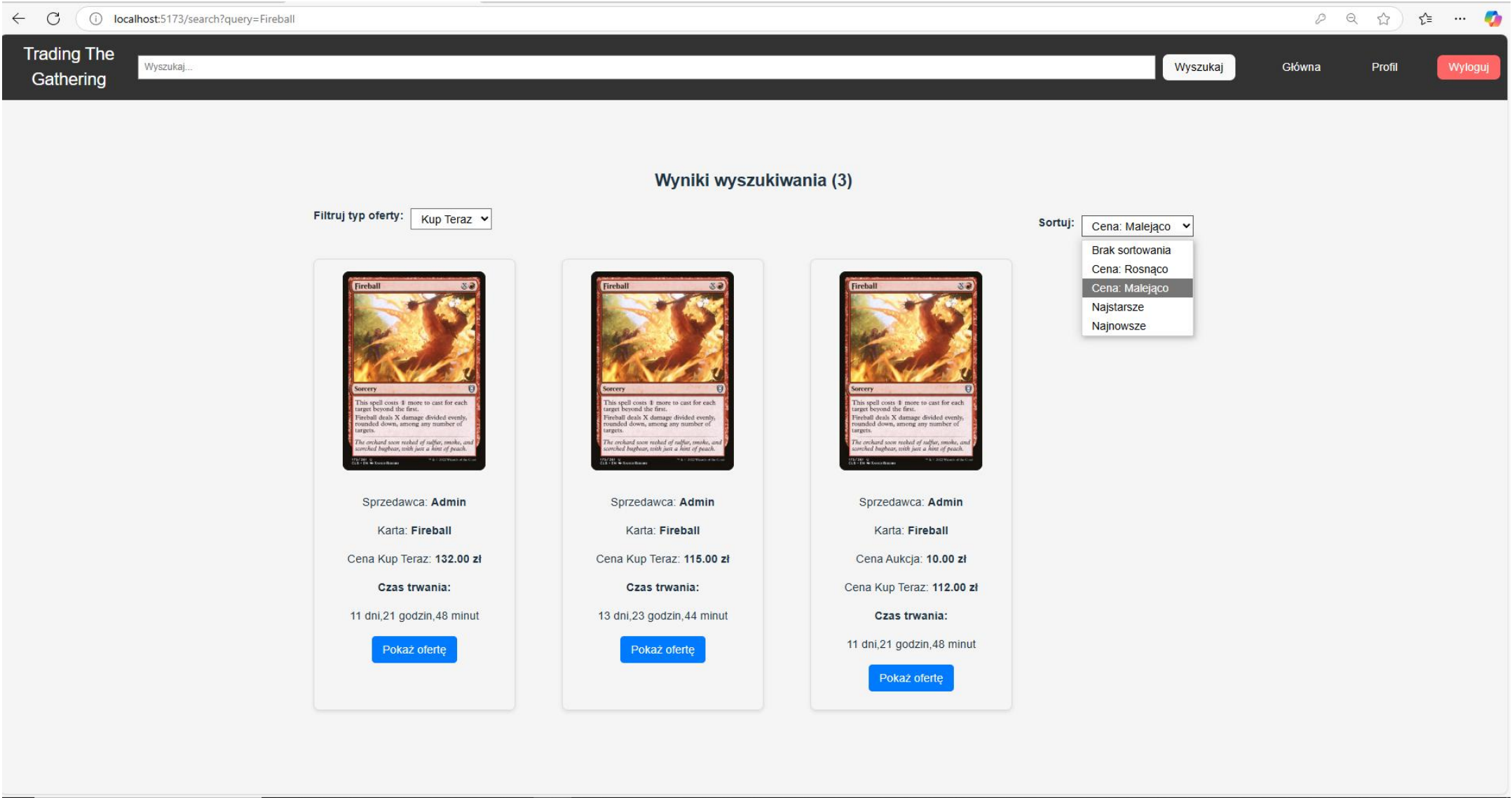
Numer konta bankowego:

[SellCard.jsx](#)

The homepage displays up to 15 latest active offers. At the top, like on every other page, there is a navigation bar. It contains a **search bar**, which, just like when creating an offer, suggests card names.



The search bar has two filters. One filters by offer type, and the other sorts by price or duration.



[SearchResults.jsx](#)

The offer overview, which also allows for purchasing or bidding on the card. The button at the bottom left allows you to view the front or back of the card.

Trading The Gathering

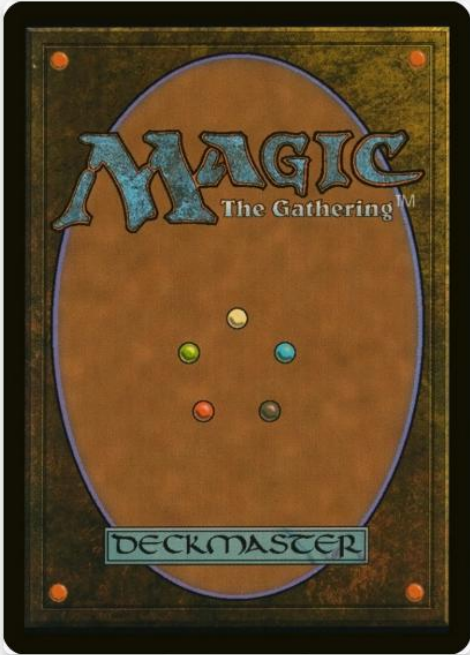
Wyszukaj

Główna

Profil

Wyloguj

Fireball



Pokaż Przód

Sprzedawca: **Admin**

Aukcja Cena: **10.00 zł**

Dodaj swoją ofertę

Kup Teraz Cena: **112.00 zł**

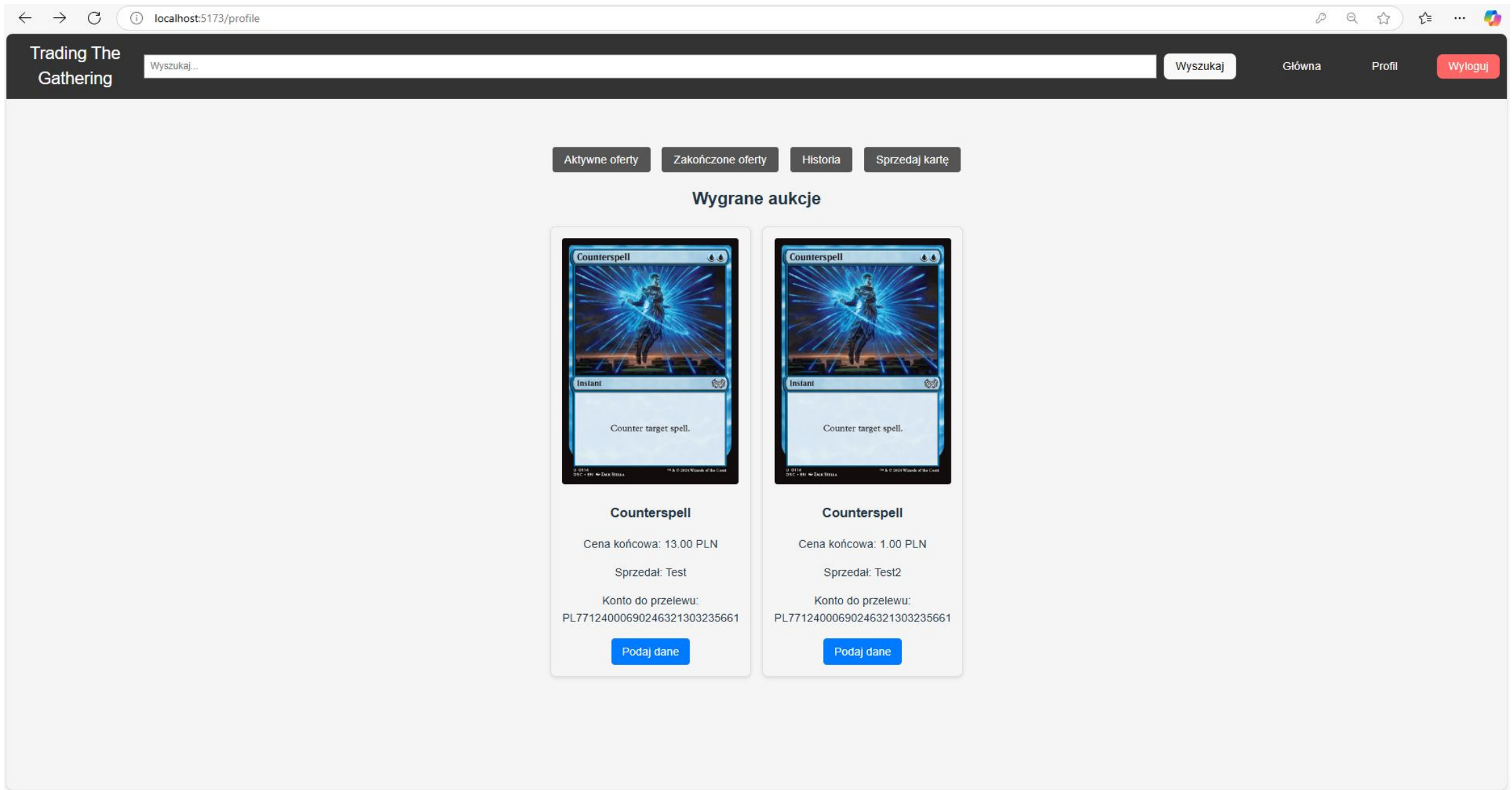
Kup Teraz

Czas trwania:

11 dni, 21 godziny, 45 minut

[OfferDetails.jsx](#)

User profile view. Unfortunately, due to lack of time, it's quite basic. It only shows the won offers. Of the four top buttons, only „Sprzedaj kartę”, works. Button redirects to the page from the "Listing cards" section.



A view where you can add your details to the shipment after winning the auction

Trading The Gathering

Wyszukaj...


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

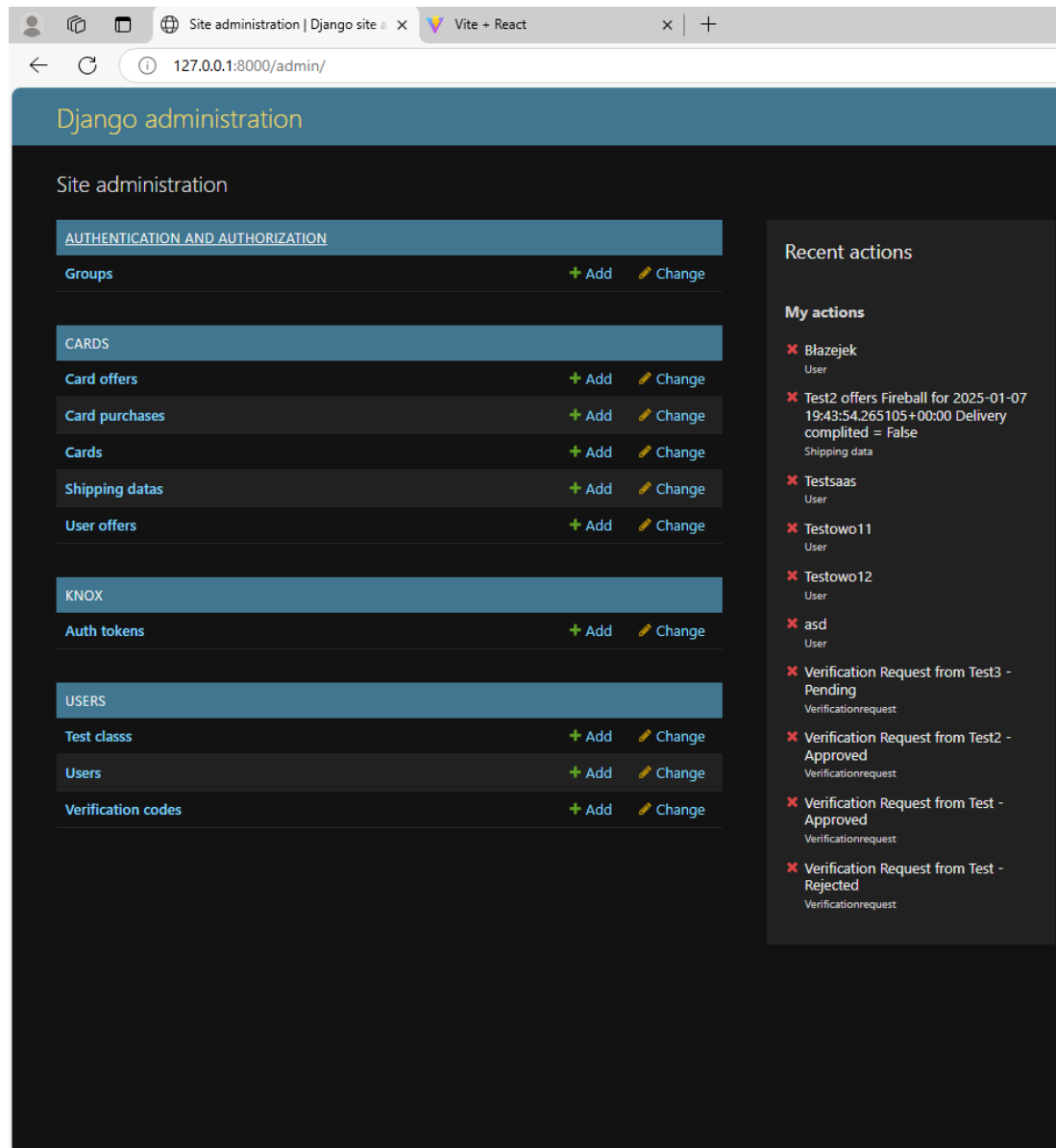
Miasto

Kod pocztowy

Zatwierdź dane do wysyłki

[Shipment.jsx](#)

View from the admin panel to the database



Card offers is a table containing card offers. In ***User offers***, records are created when a user bids on or purchases a card. Each ***User offer*** is connected to a specific ***Card offer***, just like ***Shipping data***, which stores addresses. ***Cards*** contains the cards (name along with the image).

Card purchases and ***Test classes*** are not used. They are remnants from testing.

The rest is self-explanatory. I can only add that the ***Users*** class has been modified.

More information can be found in the ***models.py*** files in the ***users*** and ***cards*** folders.

Unit tests were written at the very end. Since I worked on both the backend and frontend, I conducted tests on the fly, directly on the working (or not) website. I wrote unit tests mainly to learn how to do it. They are only present in **tests.py**.