

Projekt: Currencies

Autorzy: Kamil Świdorski 51451, Karina Tylmanowska 51426, Julia Adamska 53376, Michał Chudzik 55251, Dawid Stępniewski 36320

Link do repozytorium projektu w GitHub:

https://github.com/xBonioo/Currencies-51451_55251_53376_51426_36320

Link do iteracji projektu w Trello:

<https://trello.com/b/DpjqZnA/currencies>

O projekcie

Projekt ma na celu stworzenie wielowarstwowej aplikacji do wymiany walut online. Platforma oferuje podstawowe funkcje, takie jak możliwość wymiany walut, rejestracja nowych użytkowników, oraz logowanie do konta. Użytkownicy będą mieć również dostęp do przeglądania aktualnych oraz historycznych kursów walut, co pozwoli im śledzić zmiany i podejmować świadome decyzje w zakresie wymiany walut. W ramach projektu skupiamy się na zapewnieniu intuicyjnej obsługi oraz stabilności działania systemu, z myślą o zwiększeniu wygody korzystania z aplikacji przez naszych użytkowników.

Jak uruchomić aplikację?

Należy zainstalować na nasz komputer *'Docker Desktop'*.

W folderze *Currencies\docker-run* uruchamiamy plik *docker-build.cmd*

Po uruchomieniu zaczynają budować nam się obrazy back-end w *ASP.NET* oraz baza danych *MySQL* (jeśli jeszcze ich nie mamy).

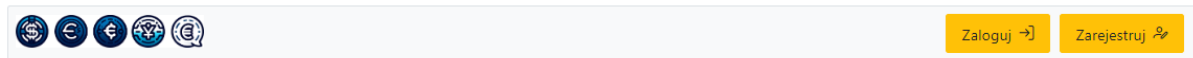
Po uruchomieniu aplikacji, Swagger można znaleźć pod linkiem:

<http://localhost:5000/swagger/index.html>

Jak korzystać z aplikacji?

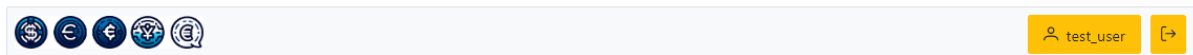
Nagłówek aplikacji:

Niezałogowany użytkownik:



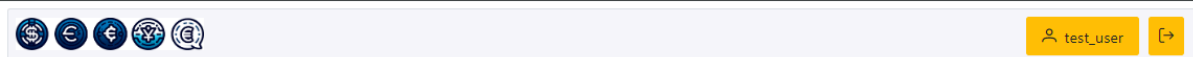
- Użytkownik ma możliwość przejść do widoku logowania lub do widoku rejestracji. Poprzez naciśnięcie na logo strony można wrócić do głównej strony (lista walut).

Zalogowany użytkownik:



Użytkownik ma możliwość wylogowania się poprzez naciśnięcie skrajnie prawego guzika, lub przejście do widoku użytkownika poprzez naciśnięcie guzika z nazwą użytkownika. Poprzez naciśnięcie na logo strony można wrócić do głównej strony (lista walut).

Lista walut wraz z kursami kupna i sprzedaży:

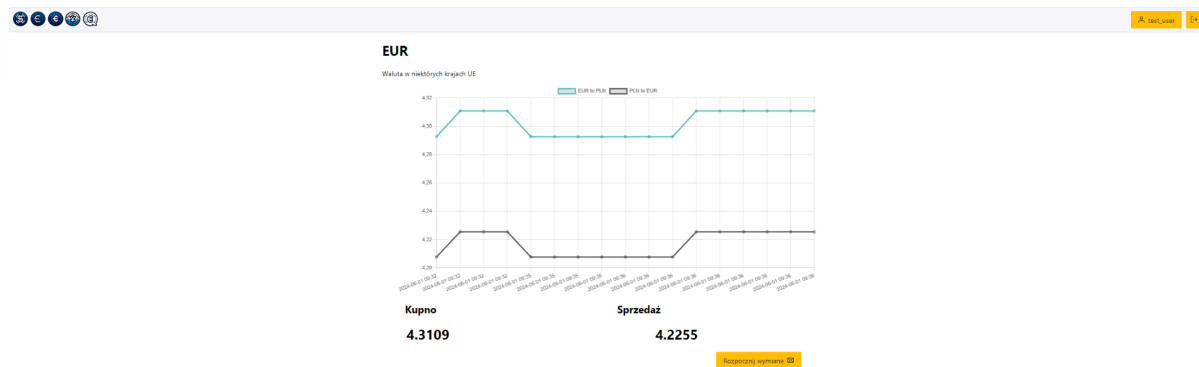


Nazwa waluty	Kod	Kupno	Sprzedaż
Dolar	USD	3.9772	3.8984
Euro	EUR	4.3109	4.2255
Funt	GBP	5.0647	4.9645
Dolar australijski	AUD	2.6362	2.584
Forint	HUF	0.011128	0.010908
Frank szwajcarski	CHF	4.3595	4.2731
Jen	JPY	0.025278	0.024778
Korona czeska	CZK	0.1744	0.171
Korona duńska	DKK	0.5778	0.5664
Korona norweska	NOK	0.3768	0.3694
Korona szwedzka	SEK	0.3743	0.3669
Dolar kanadyjski	CAD	2.9054	2.8478

Na widoku widoczna jest lista dostępnych walut z kursami kupna oraz sprzedaży.

- Użytkownik ma możliwość przejścia na widok ze szczegółami waluty poprzez podwójne kliknięcie danego wiersza waluty.

Widok z historią kursów waluty:



Na widoku wypisana jest nazwa waluty, krótki opis, wykres z historią kursów, obecny kurs kupna i sprzedaży oraz guzik otwierający okno dialogowe z wymianą walut.

- Użytkownik ma możliwość ukrywania konkretnej krzywej poprzez naciśnięcie legendy na górze wykresu.
- Użytkownik ma możliwość otworzenia okna dialogowego, gdzie może wymieniać waluty.

Okno dialogowe z wymianą walut:

Wymiana walut: Kupno EUR


Z 20 PLN Na 86.218 EUR

Anuluj Wymień

Okno wyświetla się nad widokiem z historią kursów waluty. W nagłówku okna jest opis jaką akcję użytkownik będzie wykonywać (Kupno {waluta}/Sprzedaż {waluta}). W głównej części okna znajdują się informacje o wymianie. Na dole okna guziki 'Anuluj' zamykające okno, 'Wymień' potwierdzające transakcję.

- Użytkownik ma możliwość wpisania kwoty z lewej strony którą chce wymienić.
- Użytkownik ma możliwość zamiany walut miejscami poprzez naciśnięcie przycisku z dwoma strzałkami pomiędzy polami do wpisywania walut. Czyli zamiana z kupna na sprzedaż.

Widok szczegółów konta:



test_user

[->]

Doładuj konto

Nazwa waluty	Kod	Stan konta
Polska złotówka	PLN	100 PLN
Funt	GBP	152 GBP
Dolar australijski	AUD	565 AUD

Na widoku widoczny jest przycisk do doładowania konta użytkownika oraz lista ze stanem konta w każdej walucie którą posiada użytkownik.

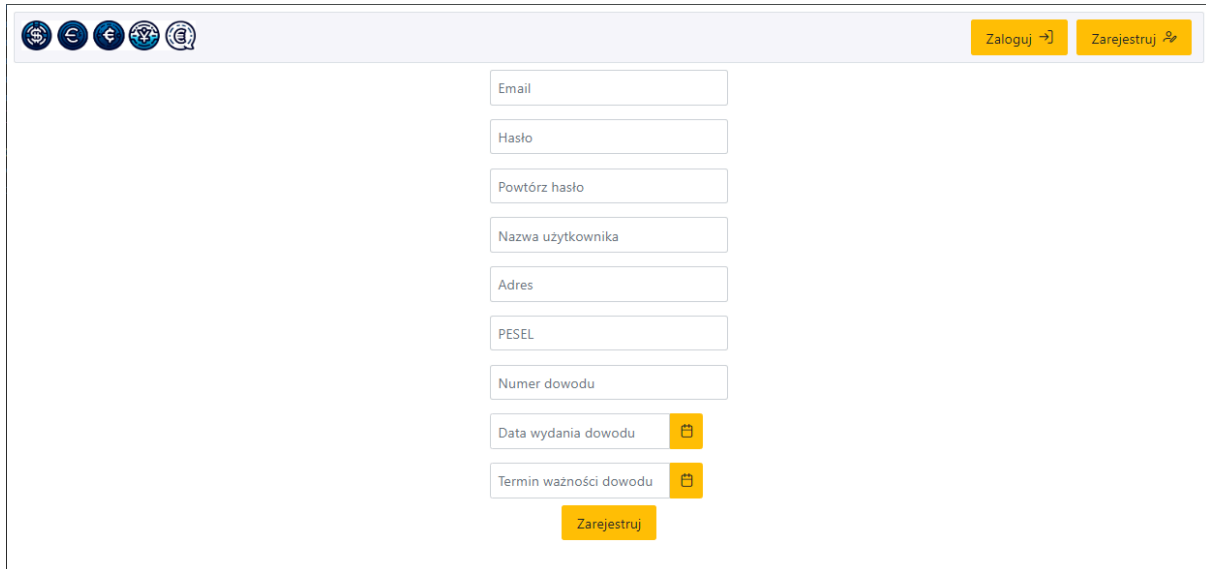
- Użytkownik ma możliwość otworzenia okna dialogowego z funkcjonalnością doładowania konta.

Okno dialogowe z doładowywaniem konta:

The image displays two screenshots of a 'Doładuj konto' (Reload account) dialog box. The top screenshot shows the dialog with an empty input field for the amount and a 'Waluta' (Currency) dropdown menu. The bottom screenshot shows the same dialog with the 'Waluta' dropdown menu open, displaying a list of currencies: USD, EUR, GBP, PLN, and AUD. The dialog box has a title 'Doładuj konto' and a question 'Ile chcesz doładować i w jakiej walucie?' (How much do you want to reload and in which currency?). There are two buttons at the bottom: 'Anuluj' (Cancel) and 'Doładuj' (Reload).

- Użytkownik ma możliwość wpisania kwoty jaką chce doładować oraz wybrać walutę w której chce zasilić konto.

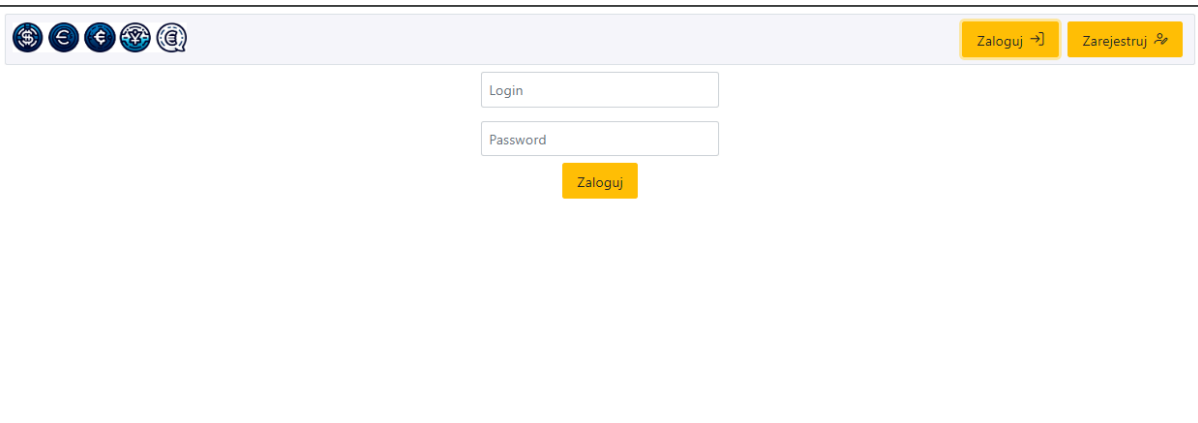
Widok rejestracji:



The registration form is displayed within a light gray header bar. On the left side of the header, there are five circular icons: a globe, a Euro symbol, a Polish Złoty symbol, a gear, and a magnifying glass. On the right side of the header, there are two yellow buttons: 'Zaloguj →' and 'Zarejestruj ↗'. The main content area contains a vertical stack of input fields: 'Email', 'Hasło', 'Powtórz hasło', 'Nazwa użytkownika', 'Adres', 'PESEL', 'Numer dowodu', 'Data wydania dowodu' (with a calendar icon), and 'Termin ważności dowodu' (with a calendar icon). A yellow 'Zarejestruj' button is positioned below the last two fields.

- Użytkownik ma możliwość zarejestrowania swojego konta poprzez wpisanie wszystkich odpowiednich danych

Widok logowania:



The login form is displayed within a light gray header bar. On the left side of the header, there are five circular icons: a globe, a Euro symbol, a Polish Złoty symbol, a gear, and a magnifying glass. On the right side of the header, there are two yellow buttons: 'Zaloguj →' and 'Zarejestruj ↗'. The main content area contains two input fields: 'Login' and 'Password'. A yellow 'Zaloguj' button is positioned below the 'Password' field.

- Użytkownik ma możliwość zalogowania się na swoje konto poprzez wpisanie poprawnego loginu i hasła

Wymagania funkcjonalne i нефункционалне

Wymagania funkcjonalne

1. Rejestracja i logowanie użytkowników

- Użytkownik może zarejestrować konto, podając adres e-mail, hasło oraz dane osobowe.
- Użytkownik może zalogować się na swoje konto, podając nazwę użytkownika i hasło.

2. Profil użytkownika

- Użytkownik może zobaczyć historię swoich transakcji.
- Użytkownik ma możliwość “doładowania” swojego konta w konkretnej walucie

3. Wymiana walut

- Użytkownik może wybrać walutę, którą chce sprzedać, oraz walutę, którą chce kupić.
- Aplikacja wyświetla kursy walut
- Aplikacja wyświetla historię kursów walut
- Użytkownik może wprowadzić kwotę, którą chce wymienić,
- Użytkownik otrzymuje wyliczoną kwotę po ustalonym kursie wymiany.
- Użytkownik może zatwierdzić transakcję wymiany walut.

Wymagania нефункционалне

1. Wydajność

- Czas odpowiedzi na zapytania użytkownika nie powinien przekraczać 2 sekund.

2. Bezpieczeństwo

- Aplikacja powinna korzystać z JWT (JSON Web Token) do autoryzacji i uwierzytelniania użytkowników.

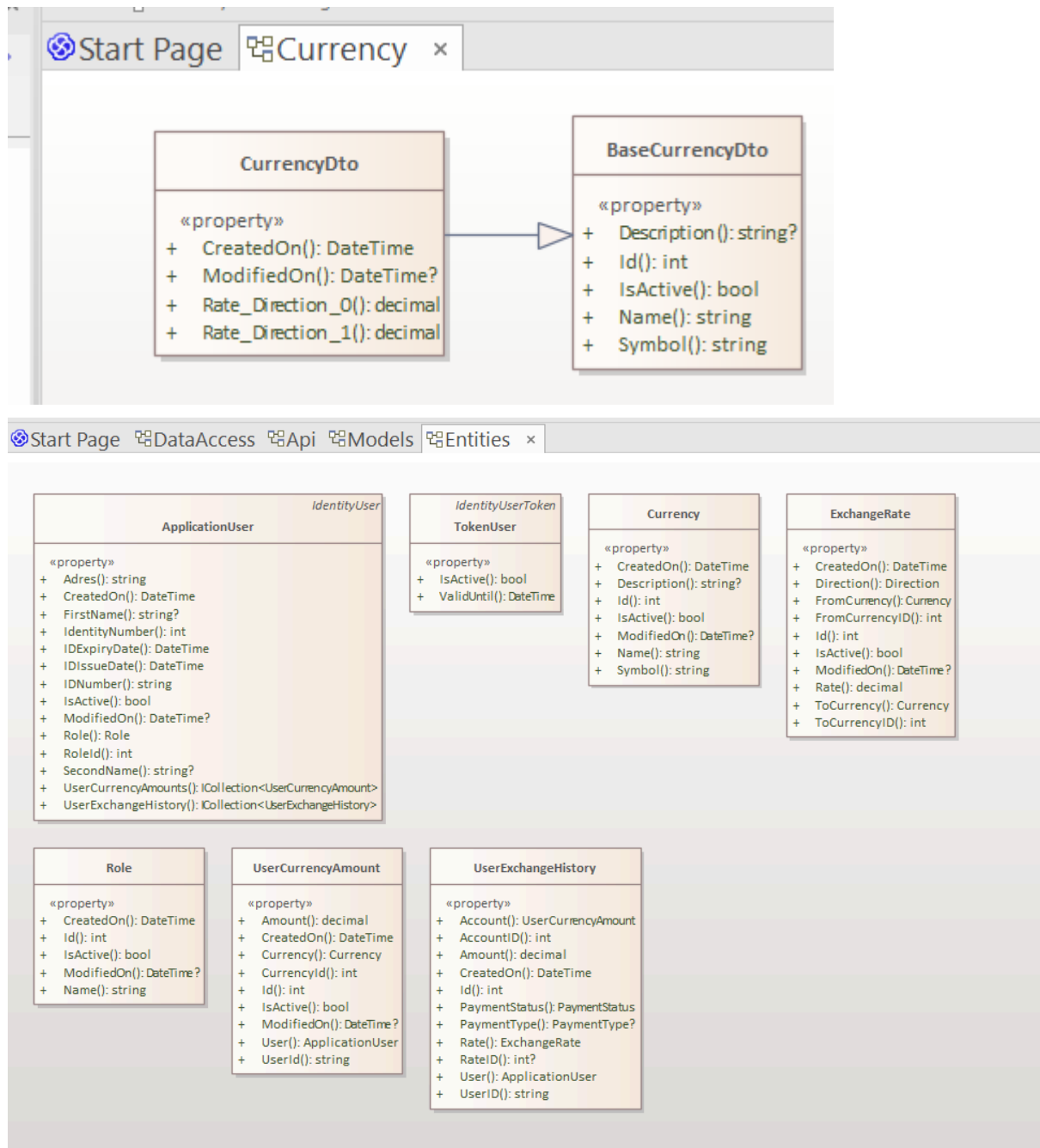
3. Użyteczność

- Interfejs użytkownika powinien być intuicyjny i łatwy w obsłudze.

Diagramy klas

Diagramy klas znajdziemy w pliku 'diagram.qea'

Plik możemy otworzyć za pomocą 'Enterprise Architect' (istnieje wersja trial na 30 dni za darmo)



Opis sposobów i metod testowania

Struktura testów

Testy jednostkowe w projekcie są zorganizowane przy użyciu frameworka *xUnit*. Testy są zgrupowane w klasy, które dziedziczą po *IClassFixture<T>* zapewniając wspólną konfigurację testów, takie jak *BaseTestFixture*. Każda klasa testowa odpowiada określonemu zestawowi funkcjonalności, na przykład kontrolerom odpowiedzialnym za różne zasoby w systemie (np. *CurrencyControllerTests*, *ExchangeRateControllerTests*).

Testy jednostkowe

Testy jednostkowe w projekcie są stosowane do weryfikacji poprawności logiki biznesowej w izolacji. Są one pisane przy użyciu frameworka *xUnit*, który pozwala na definiowanie testów za pomocą atrybutu *[Fact]*. Przykładowe metody testowe to *GetAll_Currencies_ReturnPageResult*, *GetById_Currency_ReturnCurrency*, *Create_Currency_ReturnNewCurrency* i wiele innych. Każdy test składa się z trzech głównych kroków: przygotowania danych (arrange), wykonania operacji (act) oraz weryfikacji wyników (assert).

Testowanie asynchroniczne

Wiele testów jednostkowych w projekcie jest asynchronicznych, co pozwala na testowanie operacji, które wykonują się w tle lub oczekują na dane z zewnętrznych źródeł. Metody testowe oznaczone są jako *async Task* i korzystają z mechanizmów *await*, aby zapewnić poprawne oczekiwanie na zakończenie operacji asynchronicznych przed weryfikacją wyników.

Mockowanie i dependency injection

Projekt wykorzystuje dependency injection do wstrzykiwania zależności do testowanych klas. Dzięki temu możliwe jest łatwe podmienianie rzeczywistych implementacji serwisów na mocki lub stuby w celach testowych. Do mapowania obiektów używany jest *AutoMapper*, który jest konfigurowany w konstruktorach klas testowych.

Testowanie integracyjne

Niektóre testy w projekcie mają charakter testów integracyjnych, ponieważ wykorzystują bazę danych *SQLite* w pamięci (*SqliteConnection("DataSource=:memory:"))*. Dzięki temu można testować rzeczywistą interakcję z bazą danych bez wpływu na produkcyjne dane. Przykładowo, klasy testowe dziedziczą po *BaseTestFixture*, który inicjuje bazę danych w pamięci i zapewnia kontekst (*TableContext*).

Testowanie operacji *CRUD*

Projekt zawiera testy pokrywające pełny cykl operacji *CRUD* (*Create, Read, Update, Delete*) dla różnych zasobów, takich jak waluty, kursy walut, role użytkowników, czy historię wymiany użytkowników. Każda operacja *CRUD* ma przypisany zestaw testów jednostkowych, które sprawdzają poprawność działania metod w kontrolerach i serwisach odpowiedzialnych za te operacje.

Testowanie formularzy edycyjnych

Testy jednostkowe obejmują również weryfikację poprawności formularzy do edycji dla różnych zasobów. Przykładowe testy sprawdzają, czy formularze są poprawnie zwracane, czy zawierają domyślne wartości i czy odpowiednio reagują na różne scenariusze wejściowe.

Endpoints

ExchangeRateController

1. GET */api/exchange-rate*

- Opis: Pobiera wszystkie dostępne kursy walut.
- Odpowiedzi:
 - *200 OK*: Zwraca wszystkie dostępne kursy walut.
 - *500 Internal Server Error*: Wewnętrzny błąd serwera.

2. GET */api/exchange-rate/{id}*

- Opis: Zwraca kurs waluty o podanym identyfikatorze.
- Odpowiedzi:
 - *200 OK*: Zwraca kurs waluty.
 - *404 Not Found*: Kurs waluty nie został znaleziony.

3. POST */api/exchange-rate*

- Opis: Tworzy nowy kurs waluty.
- Odpowiedzi:
 - *201 Created*: Kurs waluty został pomyślnie utworzony.
 - *400 Bad Request*: Niepoprawny obiekt JSON z parametrami.

4. GET */api/exchange-rate/{id}/edit-form*

- Opis: Pobiera formularz edycji kursu waluty.
- Odpowiedzi:
 - *200 OK*: Formularz edycji kursu waluty.
 - *404 Not Found*: Formularz edycji kursu waluty nie został znaleziony.

5. POST */api/exchange-rate/{id}/edit*

- Opis: Aktualizuje kurs waluty o podanym identyfikatorze.
- Odpowiedzi:
 - *200 OK*: Kurs waluty został pomyślnie zaktualizowany.
 - *400 Bad Request*: Niepoprawny obiekt JSON z parametrami.
 - *404 Not Found*: Kurs waluty nie został znaleziony.

6. DELETE */api/exchange-rate/{id}/delete*

- Opis: Usuwa (dezaktywuje) kurs waluty.
- Odpowiedzi:
 - *200 OK*: Kurs waluty został pomyślnie usunięty.
 - *404 Not Found*: Kurs waluty nie został znaleziony.

PaymentRateController

1. GET */api/payment-rate*

- Opis: Pobiera wszystkie dostępne kursy płatności.
 - Odpowiedzi:
 - 200 OK: Zwraca wszystkie dostępne kursy płatności.
 - 500 *Internal Server Error*: Wewnętrzny błąd serwera.
2. GET */api/payment-rate/{id}*
- Opis: Zwraca kurs płatności o podanym identyfikatorze.
 - Odpowiedzi:
 - 200 OK: Zwraca kurs płatności.
 - 404 *Not Found*: Kurs płatności nie został znaleziony.
3. POST */api/payment-rate*
- Opis: Tworzy nowy kurs płatności.
 - Odpowiedzi:
 - 201 *Created*: Kurs płatności został pomyślnie utworzony.
 - 400 *Bad Request*: Niepoprawny obiekt JSON z parametrami.
4. GET */api/payment-rate/{id}/edit-form*
- Opis: Pobiera formularz edycji kursu płatności.
 - Odpowiedzi:
 - 200 OK: Formularz edycji kursu płatności.
 - 404 *Not Found*: Formularz edycji kursu płatności nie został znaleziony.
5. POST */api/payment-rate/{id}/edit*
- Opis: Aktualizuje kurs płatności o podanym identyfikatorze.
 - Odpowiedzi:
 - 200 OK: Kurs płatności został pomyślnie zaktualizowany.
 - 400 *Bad Request*: Niepoprawny obiekt JSON z parametrami.
 - 404 *Not Found*: Kurs płatności nie został znaleziony.
6. DELETE */api/payment-rate/{id}/delete*
- Opis: Usuwa (dezaktywuje) kurs płatności.
 - Odpowiedzi:
 - 200 OK: Kurs płatności został pomyślnie usunięty.
 - 404 *Not Found*: Kurs płatności nie został znaleziony.

RoleController

1. GET */api/role*
- Opis: Pobiera wszystkie dostępne role.
 - Odpowiedzi:
 - 200 OK: Zwraca wszystkie dostępne role.
 - 500 *Internal Server Error*: Wewnętrzny błąd serwera.
2. GET */api/role/{id}*

- Opis: Zwraca rolę o podanym identyfikatorze.
- Odpowiedzi:
 - 200 OK: Zwraca rolę.
 - 404 Not Found: Rola nie została znaleziona.

3. POST */api/role*

- Opis: Tworzy nową rolę.
- Odpowiedzi:
 - 201 Created: Rola została pomyślnie utworzona.
 - 400 Bad Request: Niepoprawny obiekt JSON z parametrami.

4. GET */api/role/{id}/edit-form*

- Opis: Pobiera formularz edycji roli.
- Odpowiedzi:
 - 200 OK: Formularz edycji roli.
 - 404 Not Found: Formularz edycji roli nie został znaleziony.

5. POST */api/role/{id}/edit*

- Opis: Aktualizuje rolę o podanym identyfikatorze.
- Odpowiedzi:
 - 200 OK: Rola została pomyślnie zaktualizowana.
 - 400 Bad Request: Niepoprawny obiekt JSON z parametrami.
 - 404 Not Found: Rola nie została znaleziona.

6. DELETE */api/role/{id}/delete*

- Opis: Usuwa (dezaktywuje) rolę.
- Odpowiedzi:
 - 200 OK: Rola została pomyślnie usunięta.
 - 404 Not Found: Rola nie została znaleziona.

UserController

1. POST */api/user/register*

- Opis: Rejestruje nowego użytkownika i wysyła email potwierdzający na adres email użytkownika.
- Odpowiedzi:
 - 201 Created: Użytkownik został pomyślnie utworzony i email potwierdzający został wysłany.
 - 401 Unauthorized: Błąd związany z tokenami.
 - 500 Internal Server Error: Link potwierdzający nie mógł zostać utworzony.

2. POST */api/user/signin*

- Opis: Loguje użytkownika za pomocą nazwy użytkownika i hasła.
- Odpowiedzi:
 - 200 OK: Zalogowano pomyślnie.
 - 400 Bad Request: Nieprawidłowa nazwa użytkownika lub hasło.

- *401 Unauthorized*: Błąd związany z tokenami.
- *500 Internal Server Error*: Wewnętrzny błąd serwera.

3. POST */api/user/signout*

- Opis: Wylogowuje użytkownika, usuwając token odświeżania z bazy danych.
- Odpowiedzi:
 - *200 OK*: Wylogowano pomyślnie.
 - *401 Unauthorized*: Błąd związany z tokenami.
 - *500 Internal Server Error*: Wewnętrzny błąd serwera.

4. POST */api/user/refreshtoken*

- Opis: Generuje nowy token dostępu (JWT). Wymaga podania starego tokena dostępu w nagłówku i aktywnego tokena odświeżania w ciele żądania.
- Odpowiedzi:
 - *200 OK*: Nowy token dostępu i odświeżania.
 - *401 Unauthorized*: Błąd związany z tokenami.
 - *500 Internal Server Error*: Wewnętrzny błąd serwera.

5. GET */api/user/get-history*

- Opis: Pobiera całą historię wymiany walut użytkownika.
- Odpowiedzi:
 - *200 OK*: Zwraca historię wymiany walut użytkownika.
 - *500 Internal Server Error*: Wewnętrzny błąd serwera.

6. GET */api/user/get-history/{id}*

- Opis: Zwraca historię wymiany walut użytkownika o podanym identyfikatorze.
- Odpowiedzi:
 - *200 OK*: Zwraca historię wymiany walut użytkownika.
 - *404 Not Found*: Historia wymiany walut użytkownika nie została znaleziona.

7. POST */api/user/get-user*

- Opis: Pobiera informacje o użytkowniku.
- Odpowiedzi:
 - *200 OK*: Zwraca informacje o użytkowniku.
 - *500 Internal Server Error*: Wewnętrzny błąd serwera.

UserCurrencyAmountController

1. GET */api/user-amount*

- Opis: Pobiera wszystkie ilości walut użytkownika.
- Odpowiedzi:
 - *200 OK*: Zwraca wszystkie ilości walut użytkownika.
 - *500 Internal Server Error*: Wewnętrzny błąd serwera.

2. GET */api/user-amount/{id}*

- Opis: Zwraca ilości walut użytkownika o podanym identyfikatorze.
- Odpowiedzi:
 - *200 OK*: Zwraca ilości walut użytkownika.
 - *404 Not Found*: Ilości walut użytkownika nie zostały znalezione.

3. POST */api/user-amount/convert*

- Opis: Konwertuje ilości walut użytkownika.
- Odpowiedzi:
 - *200 OK*: Ilości walut użytkownika zostały pomyślnie skonwertowane.
 - *404 Not Found*: Ilości walut użytkownika nie zostały znalezione lub wystąpił problem.

4. GET */api/user-amount/{id}/edit-form*

- Opis: Pobiera formularz edycji ilości walut użytkownika.
- Odpowiedzi:
 - *200 OK*: Formularz edycji ilości walut użytkownika.
 - *404 Not Found*: Formularz edycji ilości walut użytkownika nie został znaleziony.

5. POST */api/user-amount/add*

- Opis: Dodaje nowe ilości walut użytkownika.
- Odpowiedzi:
 - *201 Created*: Ilości walut użytkownika zostały pomyślnie dodane.
 - *400 Bad Request*: Niepoprawny obiekt JSON z parametrami.

6. POST */api/user-amount/{id}/edit*

- Opis: Aktualizuje ilości walut użytkownika o podanym identyfikatorze.
- Odpowiedzi:
 - *200 OK*: Ilości walut użytkownika zostały pomyślnie zaktualizowane.
 - *400 Bad Request*: Niepoprawny obiekt JSON z parametrami.
 - *404 Not Found*: Ilości walut użytkownika nie zostały znalezione.

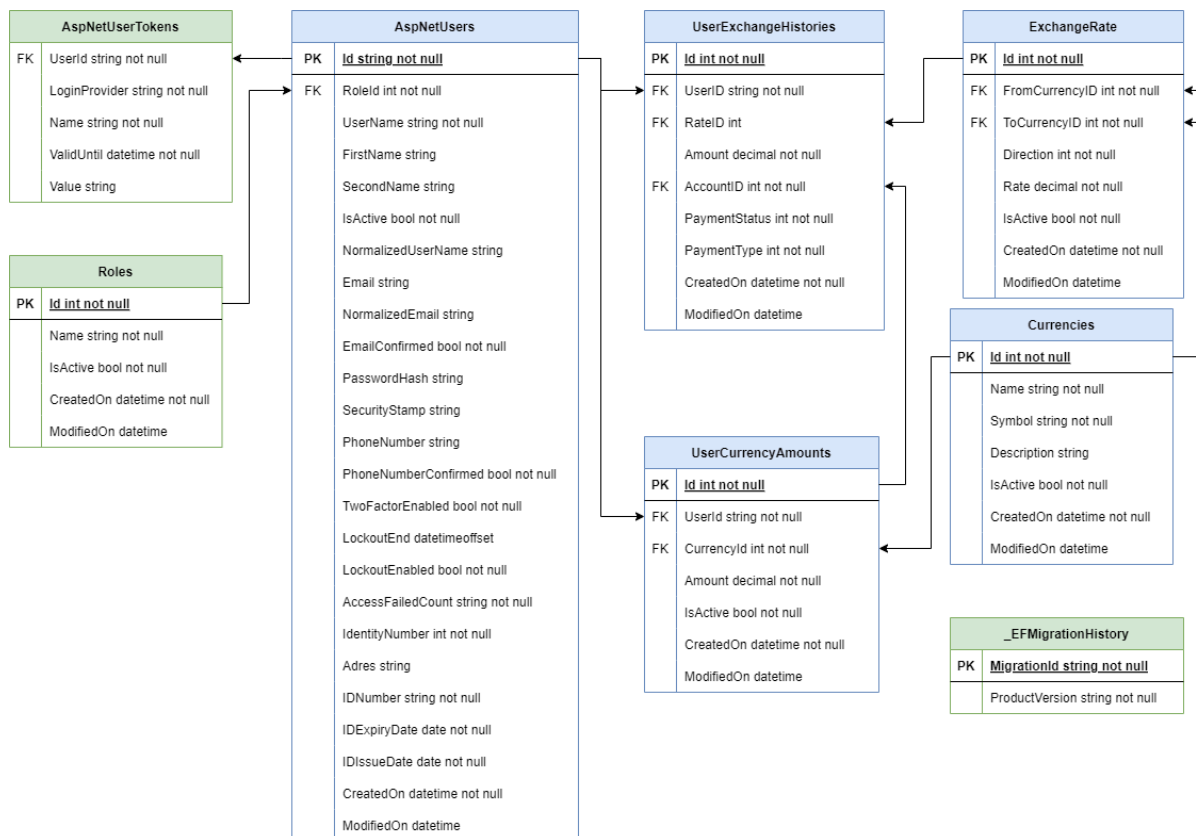
7. DELETE */api/user-amount/{id}/delete*

- Opis: Usuwa (dezaktywuje) ilości walut użytkownika.
- Odpowiedzi:
 - *200 OK*: Ilości walut użytkownika zostały pomyślnie usunięte.
 - *404 Not Found*: Ilości walut użytkownika nie zostały znalezione.

Baza danych

W tej sekcji przedstawiamy strukturę oraz funkcje bazy danych, która wspiera działanie naszej platformy do wymiany walut. Baza danych jest oparta na relacyjnym modelu danych i zapewnia niezbędną funkcjonalność do przechowywania informacji o użytkownikach, transakcjach oraz kursach walut.

Model bazy danych



Opis tabel w bazie

Tabela: AspNetUsers

- Przechowuje informacje o użytkownikach aplikacji.
- Kolumny:
 - Id: Klucz główny identyfikujący użytkownika.
 - RoleId: Identyfikator roli przypisanej do użytkownika (Klucz obcy).
 - UserName: Nazwa użytkownika.
 - FirstName: Imię użytkownika.
 - SecondName: Nazwisko użytkownika.
 - IsActive: Określa czy konto użytkownika jest aktywne.
 - NormalizedUserName: Znormalizowana nazwa użytkownika.
 - Email: Adres e-mail użytkownika.

- ix. NormalizedEmail: Znormalizowany adres e-mail użytkownika.
- x. EmailConfirmed: Określa czy adres e-mail użytkownika został potwierdzony.
- xi. PasswordHash: Skrócona wartość hasła użytkownika.
- xii. SecurityStamp: Unikalny znacznik zabezpieczeń.
- xiii. PhoneNumber: Numer telefonu użytkownika.
- xiv. PhoneNumberConfirmed: Określa czy numer telefonu użytkownika został potwierdzony.
- xv. TwoFactorEnabled: Określa czy uwierzytelnianie dwuskładnikowe jest włączone dla użytkownika.
- xvi. LockoutEnd: Blokada konta użytkownika
- xvii. LockoutEnabled: Określa czy blokada konta użytkownika jest włączona.
- xviii. AccessFailedCount: Licznik nieudanych prób logowania.
- xix. IdentityNumber: Numer pesel użytkownika.
- xx. Adres: Adres zamieszkania użytkownika.
- xxi. IDNumber: Numer dokumentu tożsamości użytkownika.
- xxii. IDExpiryDate: Data wygaśnięcia dokumentu tożsamości.
- xxiii. IDIssueDate: Data wydania dokumentu tożsamości.
- xxiv. CreatedOn: Data utworzenia rekordu w bazie danych.
- xxv. ModifiedOn: Data ostatniej modyfikacji rekordu.

Tabela: Currencies

- Przechowuje informacje o dostępnych walutach.
- Kolumny:
 - i. Id: Klucz główny identyfikujący walutę.
 - ii. Name: Nazwa waluty.
 - iii. Symbol: Symbol waluty.
 - iv. Description: Opis waluty.
 - v. IsActive: Określa czy waluta jest aktywna.
 - vi. CreatedOn: Data utworzenia rekordu w bazie danych.
 - vii. ModifiedOn: Data ostatniej modyfikacji rekordu.

Tabela: ExchangeRate

- Przechowuje dane o wysokości kursów walut
- Tabela jest aktualizowana codziennie i przechowuje informacje historyczne
- Kolumny:
 - i. Id: Klucz główny identyfikujący kurs wymiany.
 - ii. FromCurrencyID: Identyfikator waluty źródłowej (Klucz obcy).
 - iii. ToCurrencyID: Identyfikator waluty docelowej (Klucz obcy).
 - iv. Rate: Kurs wymiany między walutami.
 - v. Direction: Określa kierunek wymiany (kupno lub sprzedaż).
 - vi. IsActive: Określa czy kurs wymiany jest aktywny.
 - vii. CreatedOn: Data utworzenia rekordu w bazie danych.
 - viii. ModifiedOn: Data ostatniej modyfikacji rekordu.

Tabela: Roles

- Przechowuje informacje o rolach użytkowników.
- Kolumny:
 - i. Id: Klucz główny identyfikujący rolę.
 - ii. Name: Nazwa roli.
 - iii. IsActive: Określa czy rola jest aktywna.
 - iv. CreatedOn: Data utworzenia rekordu w bazie danych.
 - v. ModifiedOn: Data ostatniej modyfikacji rekordu.

Tabela: UserCurrencyAmounts

- Przechowuje informacje o ilości poszczególnych walut i stanie konta użytkowników.
- Kolumny:
 - i. Id: Klucz główny identyfikujący ilość waluty użytkownika.
 - ii. UserId: Identyfikator użytkownika (Klucz obcy).
 - iii. CurrencyId: Identyfikator waluty (Klucz obcy).
 - iv. Amount: Ilość danej waluty w posiadaniu użytkownika.
 - v. IsActive: Określa czy ilość waluty jest aktywna.
 - vi. CreatedOn: Data utworzenia rekordu w bazie danych.
 - vii. ModifiedOn: Data ostatniej modyfikacji rekordu.

Tabela: UserExchangeHistories

- Przechowuje historię transakcji wymiany walut użytkowników.
- Nowy rekord dodawany jest przy każdej wymianie waluty
- Kolumny:
 - i. Id: Klucz główny identyfikujący historię wymiany użytkownika.
 - ii. UserID: Identyfikator użytkownika (Klucz obcy).
 - iii. RateID: Identyfikator kursu wymiany (Klucz obcy).
 - iv. Amount: Ilość wymienionej waluty.
 - v. AccountID: Identyfikator konta użytkownika (Klucz obcy).
 - vi. PaymentStatus: Status płatności.
 - vii. PaymentType: Typ płatności.
 - viii. CreatedOn: Data utworzenia rekordu w bazie danych.
 - ix. ModifiedOn: Data ostatniej modyfikacji rekordu.

Tabela: AspNetUserTokens

- Przechowuje dane tokenu użytkownika oraz datę jego ważności.
- Kolumny:
 - i. UserId: Identyfikator użytkownika.
 - ii. LoginProvider: Dostawca logowania, który wygenerował token.
 - iii. Name: Nazwa tokenu.
 - iv. ValidUntil: Data ważności tokenu użytkownika.
 - v. Value: Wartość tokenu.