Dokumentacja techniczna

Flash Card App

Bartosz Łukowski 138220, Miłosz Stachowiak 138417

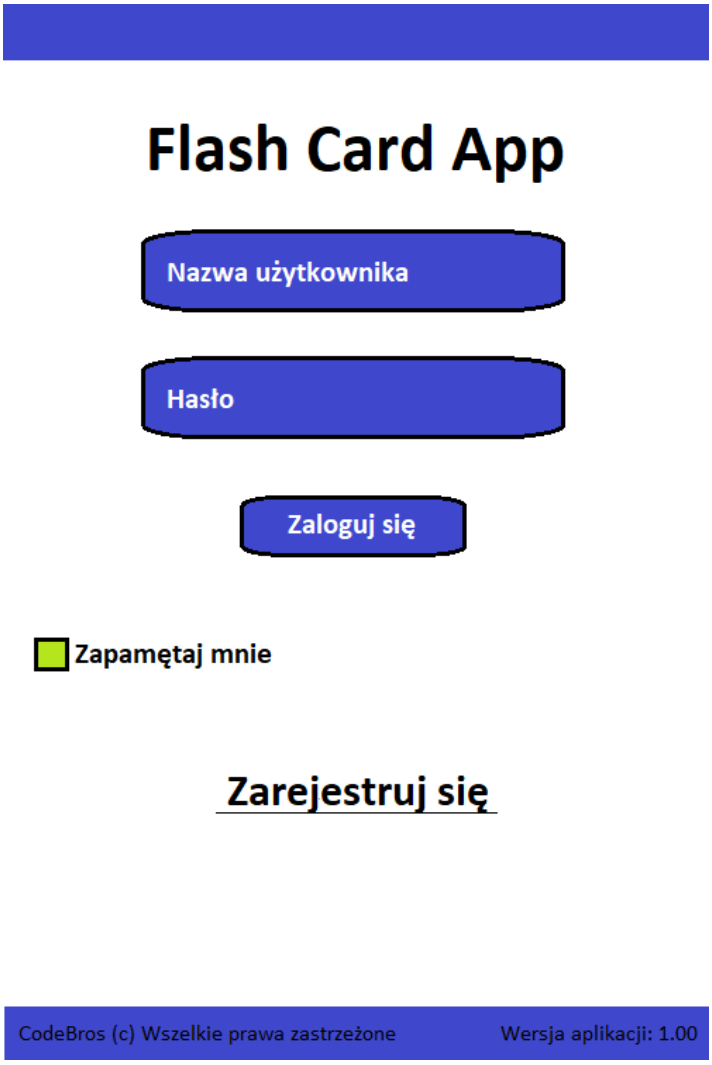
1. Pokrótce o aplikacji:

Głównym założeniem projektu jest stworzenie aplikacji mobilnej na urządzenia z systemem Android z wykorzystaniem Visual Studio 2022, bazy danych MySQL obsługiwanej poprzez **API** oraz opartej na modelu **MVVM**. Aplikacja skierowana będzie dla użytkowników chcących nauczyć się języka obcego przy pomocy **fiszek**. W aplikacji tworzyć będziemy fiszki i korzystać z nich identycznie jak w wersji “papierowej” z tą różnicą, że fiszki w aplikacji będziemy mieli zawsze pod ręką, dzielić się nimi będziemy mogli z innymi a do tworzenia ich wystarczy nam tylko chwila wolnego czasu. Zachęcić ma to do nauki dzieci oraz zapewnić bardziej ekologiczne podejście oszczędzając papier. Będzie to możliwe dzięki rozbudowanym funkcjom:
- wyboru wyświetlanych języków obcych
- tworzenia fiszek wraz z opcją wyświetlania ich w wersji flip-card(obracanej)
- grupowania fiszek po kategoriach
- dodawania/pobierania katalogów fiszek od innych użytkowników
- zapisywania w bazie lokalnej lub udostępniania katalogów fiszek innym
- usuwania, edytowania fiszek jak i całych katalogów
- implementacji intuicyjnego paska TabBar na dole ekranu

Ponadto aplikacja wymagać będzie utworzenia własnego konta poprzez rejestracje w celu zapamiętania danych i postępów użytkownika oraz umożliwi dodawanie fiszek innych użytkowników. Dodatkowo każdy użytkownik posiadać będzie opcje personalizacji i edycji własnego konta np. poprzez dodanie awataru czy zmiany hasła. Edytowalna będzie również sama aplikacja, która np. pozwoli na dodanie własnego tła fiszek. Hasło do konta będzie szyfrowane.

2. Widok poszczególnych okien użytkownika (opis systemu) oraz zaimplementowane funkcjonalności:

Okno startowe/ logowania i rejestracji
(niewidoczne po logowaniu oraz użyciu checkbox „Zapamiętaj mnie”)



Znajdziemy tutaj:

- Label z nazwą aplikacji
- Pole logowania dla nazwy użytkownika z ograniczeniem liczby znaków
- Pole logowania dla hasła użytkownika z ograniczeniem liczby znaków
- Text click „Pokaż hasło”
- Text click „Nie pamiętam hasła”
- Przycisk logowania, który sprawdzać będzie dane logowania w bazie danych
- Checkbox „Zapamiętaj mnie” dla automatycznego logowania poprzez zapamiętane dane
- Przycisk „Zarejestruj się”, który przeniesie użytkownika do okna rejestracji

Okno rejestracji
(wchodzimy poprzez przycisk „Zarejestruj się”)

Znajdziemy tutaj:

- Label „Rejestracja”
- Pola textbox wymagane i opcjonalne podczas rejestracji z ograniczeniem znaków tj. „Imię”, „Nazwisko”, „Login”, „Hasło”, „Powtórz hasło”, „Adres e-mail”
- Listbox z wyborem języka ojczystego (wymagane) czyli domyślny front fiszek
- Podgląd oraz przycisk dodawania Avatara (opcjonalne)
- Przycisk „Zarejestruj”
- Text click „Anuluj”
- Label „Pola oznaczone gwiazdką (*) są wymagane”
- Checkbox „* Oświadczam, że zapoznałem(-am) się i akceptuje treść regulaminu”
- Po kliknięciu w „regulaminu” dodatkowe okienko z regulaminem
- Okienko prawidłowej rejestracji „Witaj! [miejsce na imię] Twoja rejestracja przebiegła poprawnie.

Regulamin:

(nieprzestrzeganie może skutkować uśnięciem konta)

- Treści rasistowskie lub wulgaryzmy w loginie - zabronione
- Avatar przedstawiający treści wrażliwe lub nieodpowiednie wiekowo - zabroniony
- Rozpowszechnianie lub kopiowanie aplikacji - niedozwolone
- Wykorzystywanie aplikacji w celach innych niż prywatnych - niedozwolone

Okienko poprawnej rejestracji:

Witaj! Imię

Twoja rejestracja przebiegła poprawnie

OK

Okno główne
(widoczne po zalogowaniu oraz z którego przechodzimy do kolejnych okien)

Znajdziemy tutaj:

- Avatar użytkownika, po którego naciśnięciu wysunie się pole z opcją edycji profilu oraz przyciskiem wyloguj
- Label z napisem „Moje Fiszki”
- Przycisk dodawania omówiony poniżej
- Lista kontenerów z fiszkami posiadająca nazwę kategorii oraz kierunek tłumaczenia (flaga zamiast nazwy).
- Po przytrzymaniu grupy wysunie się okienko z opcją usunięcia grupy oraz dodania fiszek do już istniejącej kategorii
- Przycisk ikona Wyszukaj która uruchomi nowe okno do wyszukiwania fiszek
- Przycisk ikona Home do powrotu do głównego okna Moje Fiszki
- Przycisk ikona Profil która uruchomi okno z profilem i jego ustawieniami

Ponadto znajdzie się tam również przycisk do udostępnienia kategorii innym użytkownikom

Kliknięcie w kategorię uruchomi nowe okno do nauki fiszek

Przyciski dodawania:

Po kliknięciu w przycisk „+ Dodaj nową” uruchomi nam się okno dodawania. Znajdzie się tam opcja wyboru języka, poziomu zaawansowania fiszek oraz miejsce do wpisania nazwy folderu/kontenera z fiszkami inaczej zwanego kategorią. Będą to pola wymagane. Dodatkowo będzie tam opcjonalne miejsce do wpisania słówka oraz jego tłumaczenia wraz z przyciskiem „dodaj kolejny” który za każdym razem tworzyć będzie nowe pole do wpisania kolejnej fiszki. To rozwiązanie pozwoli użytkownikowi na stworzenie samego pustego kontenera lub od razu uzupełnionego o fiszki.

Opcja dodawania fiszki wraz z przyciskiem „dodaj kolejny” będzie również dostępna po przytrzymaniu istniejącej kategorii i wybraniu ikony dodawania. Pola wyboru języka, poziomu zaawansowania oraz nazwy folderu będą wtedy edytowalne i autouzupełnione. Na dole tworzyć będzie nam się również lista już dodanych fiszek.

Okno profilu
(wywołane poprzez naciśnięcie na ikonę profilu w TabBar)

Znajdziemy tutaj:

- Podgląd awatara
- Label imię i nazwisko
- Listbox z wyborem języka z bazy danych języków
- Lista wybranych języków
- Text click „Wyloguj się”

Okno Wyszukaj
(uruchamiane przyciskiem lupy z paska dolnego)

Znajdziemy tutaj:

- Avatar użytkownika, po którego naciśnięciu wysunie się pole z opcją edycji profilu oraz przyciskiem wyloguj
- Label z napisem „Wyszukiwarka”
- Pole do wyszukiwania fiszek po kategorii ograniczone do 50 znaków
- Pole wyboru poziomu językowego fiszek
- Pole wyszukiwania fiszek użytkownika ograniczone do 50 znaków
- Listbox wyboru języka fiszek (na liście znajdą się tylko te języki, które wybraliśmy w profilu)
- Grupy fiszek spełniające wymagania wyszukiwania (opisane poniżej)
- Po przytrzymaniu grupy wysunie się okienko z opcją usunięcia grupy oraz dodania fiszek do już istniejącej kategorii
- Przycisk ikona Wyszukaj która uruchomi nowe okno do wyszukiwania fiszek
- Przycisk ikona Home do powrotu do głównego okna Moje Fiszki
- Przycisk ikona Profil która uruchomi okno z profilem i jego ustawieniami

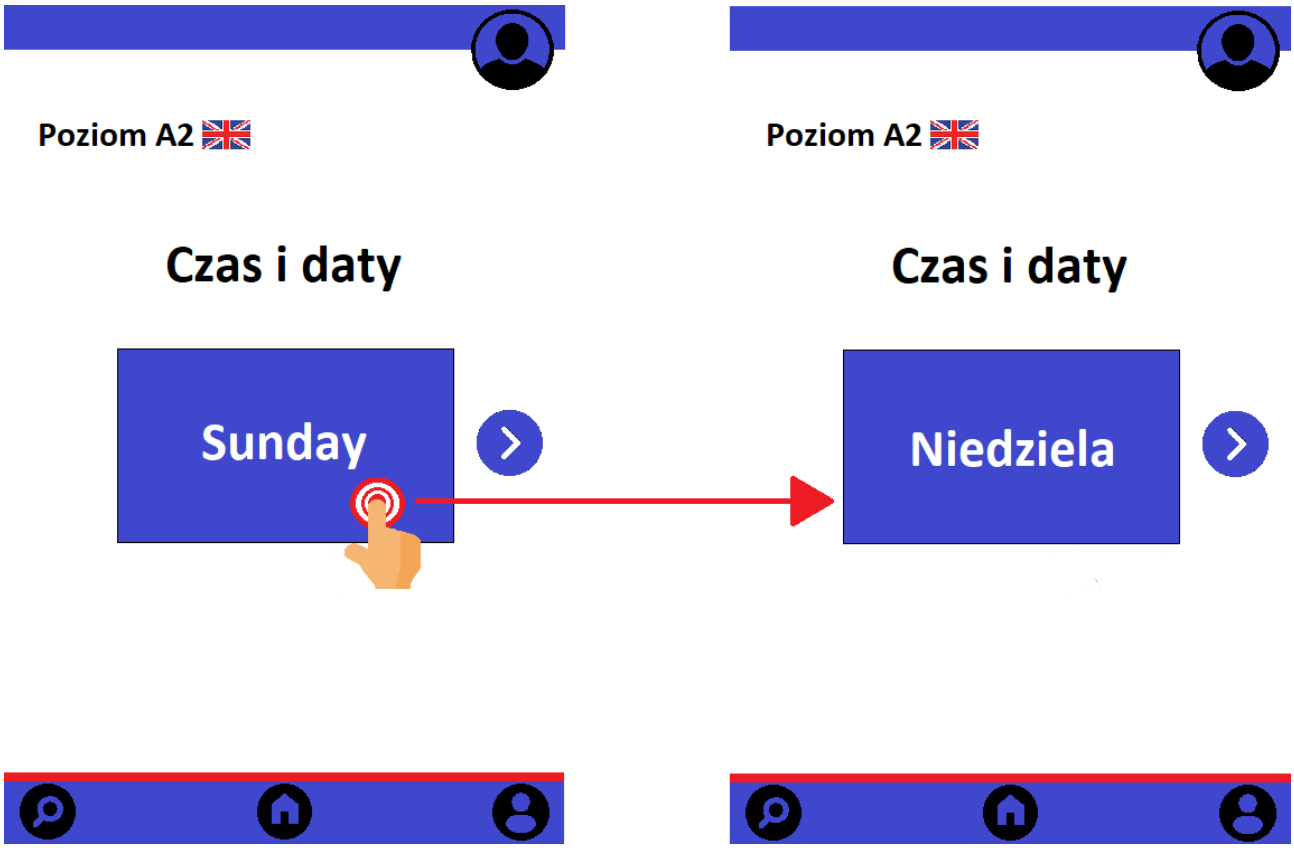
Wszystko to daje nam wiele kombinacji wyszukiwania. Dla przykładu:

- Wyszukiwanie według słów kluczowych (kategorii)
- Wyszukiwanie według poziomu językowego
- Wyszukiwanie fiszek użytkownika bez względu na poziom
- Wyszukiwanie fiszek z całej bazy tylko według języka

Wszystkie te opcje można dowolnie łączyć. Język fiszek musi być zawsze wybrany.

Okno Nauki
(uruchamiane poprzez kliknięcie w kategorię na stronie głównej)

- Flip-card: Pole z fiszką, która obraca się po naciśnięciu ujawniając tłumaczenie przypisane do fiszki.



W Flip-card znajdziemy:

- Avatar użytkownika, po którego naciśnięciu wysunie się pole z opcją edycji profilu oraz przyciskiem wyloguj
- Label z flagą informujący o poziomie językowym oraz języku nauki
- Label kategorii
- Pole z obracaną Fiszką (na odwrocie po dotknięciu widzimy tłumaczenie)
- Przycisk przejścia do następnej fiszki (automatycznie zalicza postęp jako nie umiem)

- Przycisk ikona Wyszukaj która uruchomi nowe okno do wyszukiwania fiszek
- Przycisk ikona Home do powrotu do głównego okna Moje Fiszki
- Przycisk ikona Profil która uruchomi okno z profilem i jego ustawieniami

Okno Ustawienia
(uruchamiane przyciskiem ustawień z wysuwanego paska bocznego)

Znajdziemy tutaj:

- Avatar użytkownika, po którego naciśnięciu wysunie się pole z opcją edycji profilu oraz przyciskiem wyloguj
- Label z napisem „Ustawienia”
- Wybór koloru tła fiszek
- Wybór koloru czcionki fiszek
- Text click umożliwiający wyczyszczenie wszystkich fiszek
- Text click uruchamiający okienko regulaminu
- Label z informacją o wersji aplikacji

- Pole opisujące aplikacje i jej twórców
- Przycisk ikona Wyszukaj która uruchomi nowe okno do wyszukiwania fiszek
- Przycisk ikona Home do powrotu do głównego okna Moje Fiszki
- Przycisk ikona Profil która uruchomi okno z profilem i jego ustawieniami

Lista boczna rozsuwana
(pokazująca się z lewej stronie po kliknięciu w przycisk)

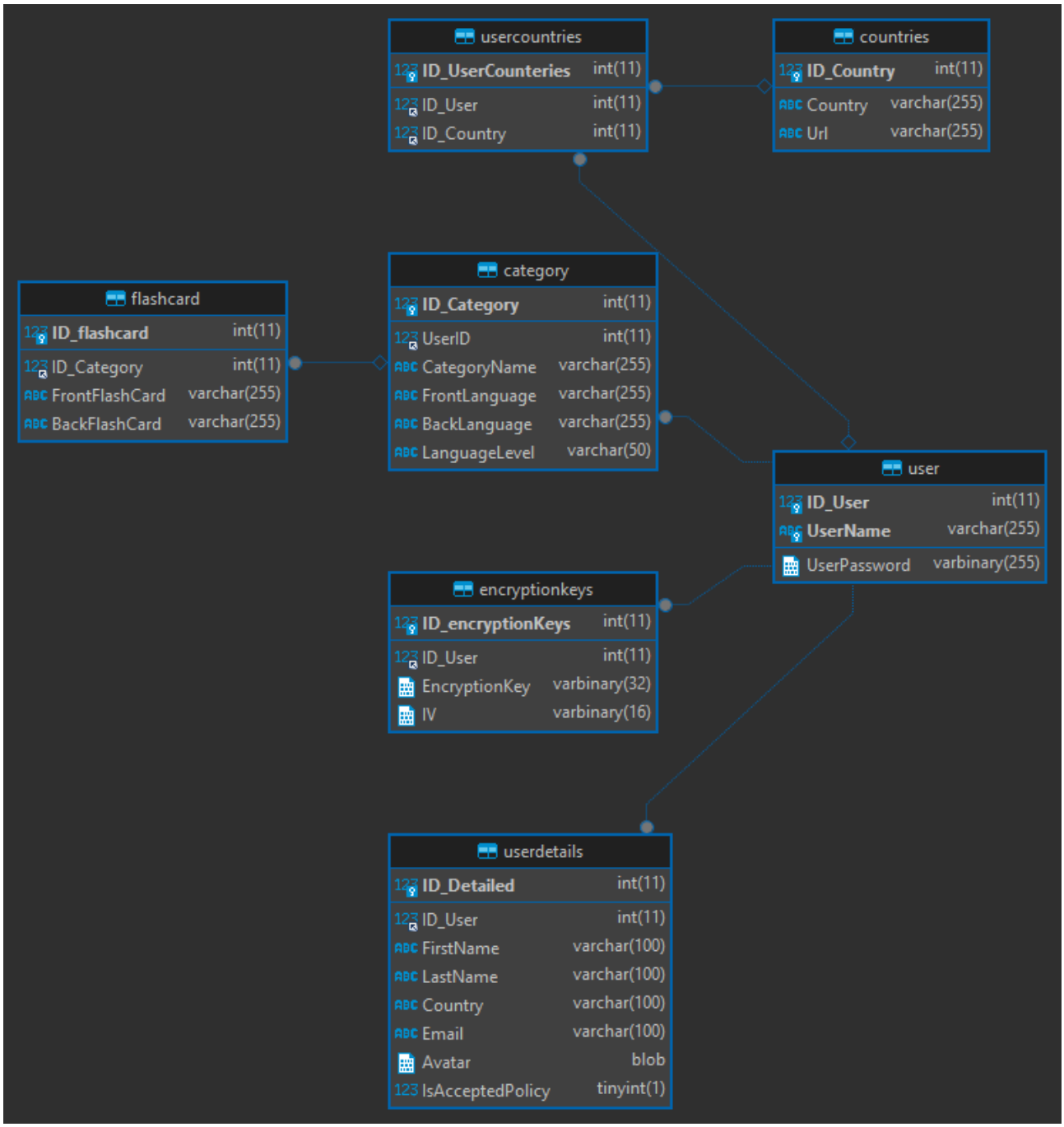
Znajdziemy tutaj:

- Przyciski dodane do paska TabBar
- Przycisk Ustawienia kierujący nas do okna ustawień aplikacji
- Przycisk Wyloguj się
- Podczas tworzenia aplikacji trafią również do tego paska inne przydatne przyciski

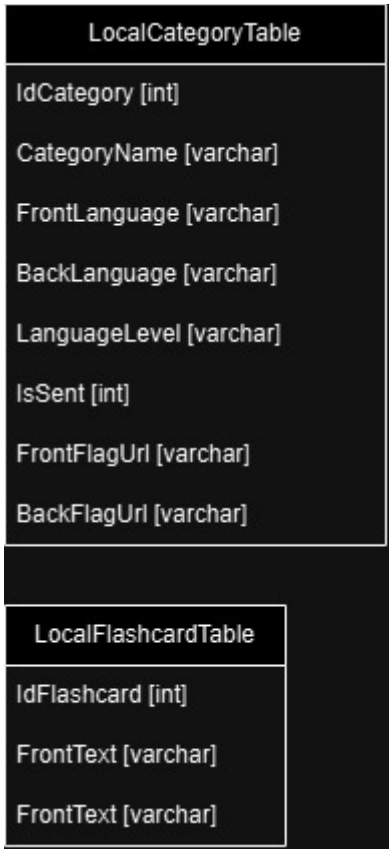
3. Baza danych:

Aplikacja posiadać będzie dwie bazy danych:

- Widoczną na poniższym schemacie bazę MySQL na serwerze zewnętrznym dla wszystkich danych użytkownika oraz udostępnionych kategorii fiszek. Na potrzeby projektu baza przeniesiona na SQL Server Express.



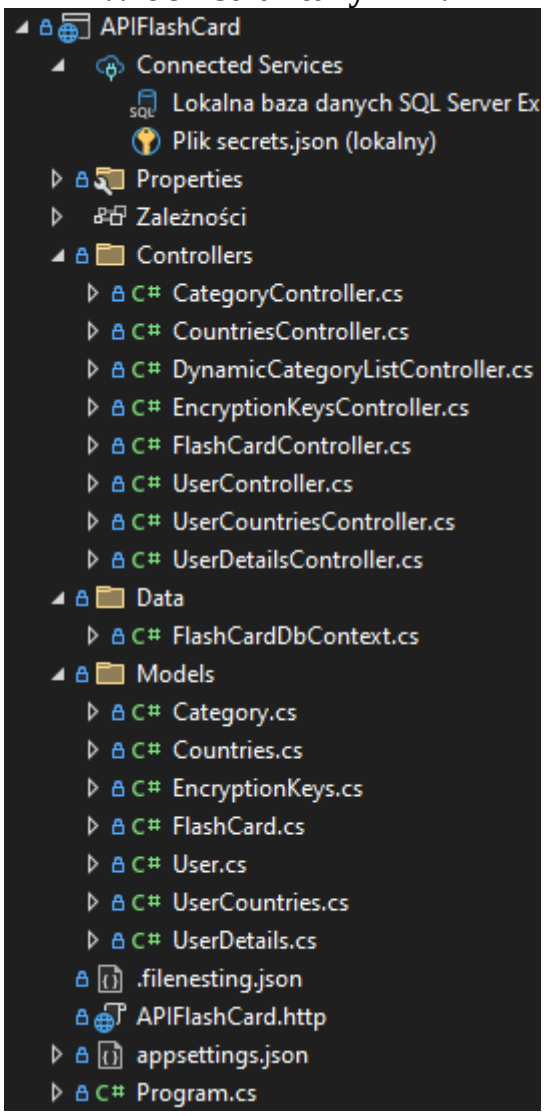
- Lokalną SQLite dla własnych kategorii fiszek, które będzie można dowolnie edytować czy usuwać.



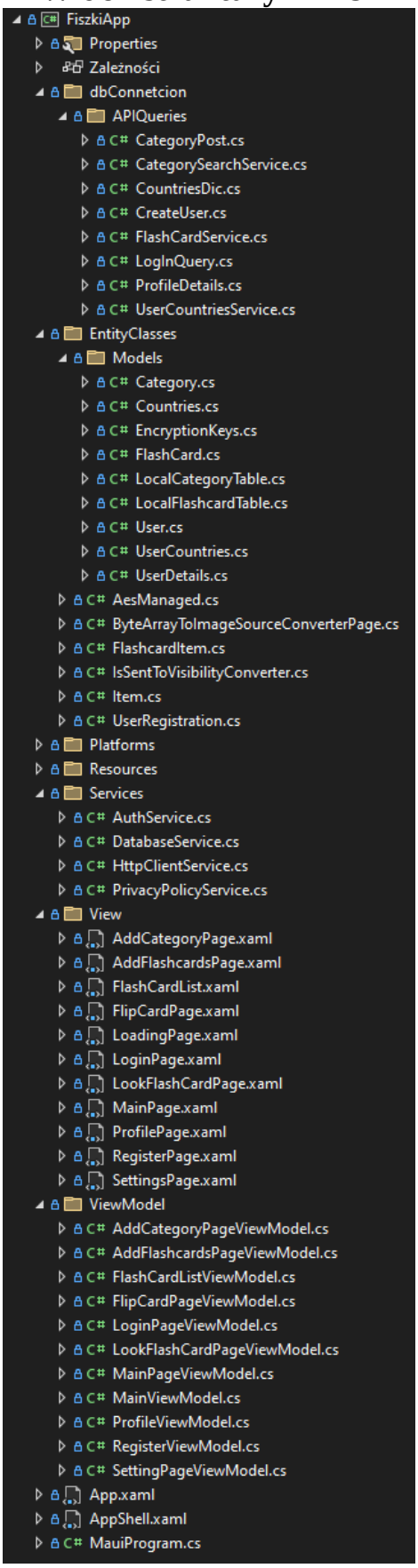
Udostępnić kategorię można tylko raz i opcja ta wtedy znika, a odpowiada za to IsSent. Usunięcie kategorii nie usunie jej z listy kategorii udostępnionych innym użytkownikom. Usunięcie wszystkich danych użytkownika w oknie "Ustawienia" wymazuje wszystkie dane z bazy lokalnej. Baza lokalna zawierać będzie tylko tabele kategorii oraz fiszek.

4. Struktura projektu:

Widok struktury API:



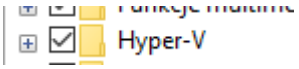
Widok struktury MAUI



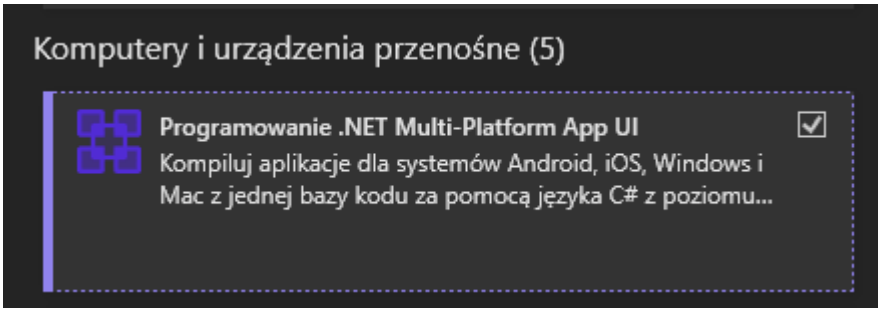
5. Instrukcja pierwszego uruchomienia aplikacji:

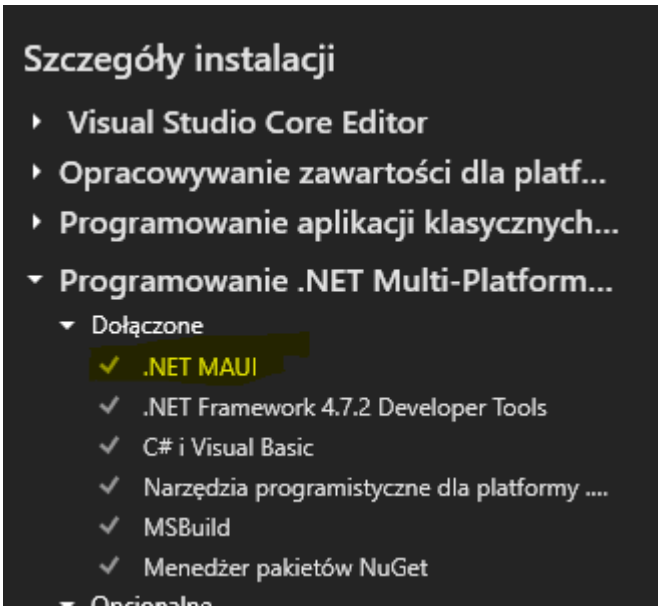
1) Należy włączyć funkcje w Windowsie Hyper-V aby to zrobić trzeba wejść w „Włącz lub wyłącz funkcje systemu Windows” następnie w rozwijanym drzewku znaleźć opcję
Hyper-V.

   Poprawnie włączona funkcja Hyper-V:

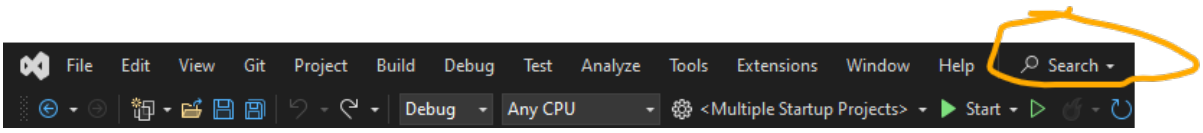


2) Następnie należy w „Visual Studio Installer” pobrać opcję „Programowanie .Net Multi-Platform App UI”:

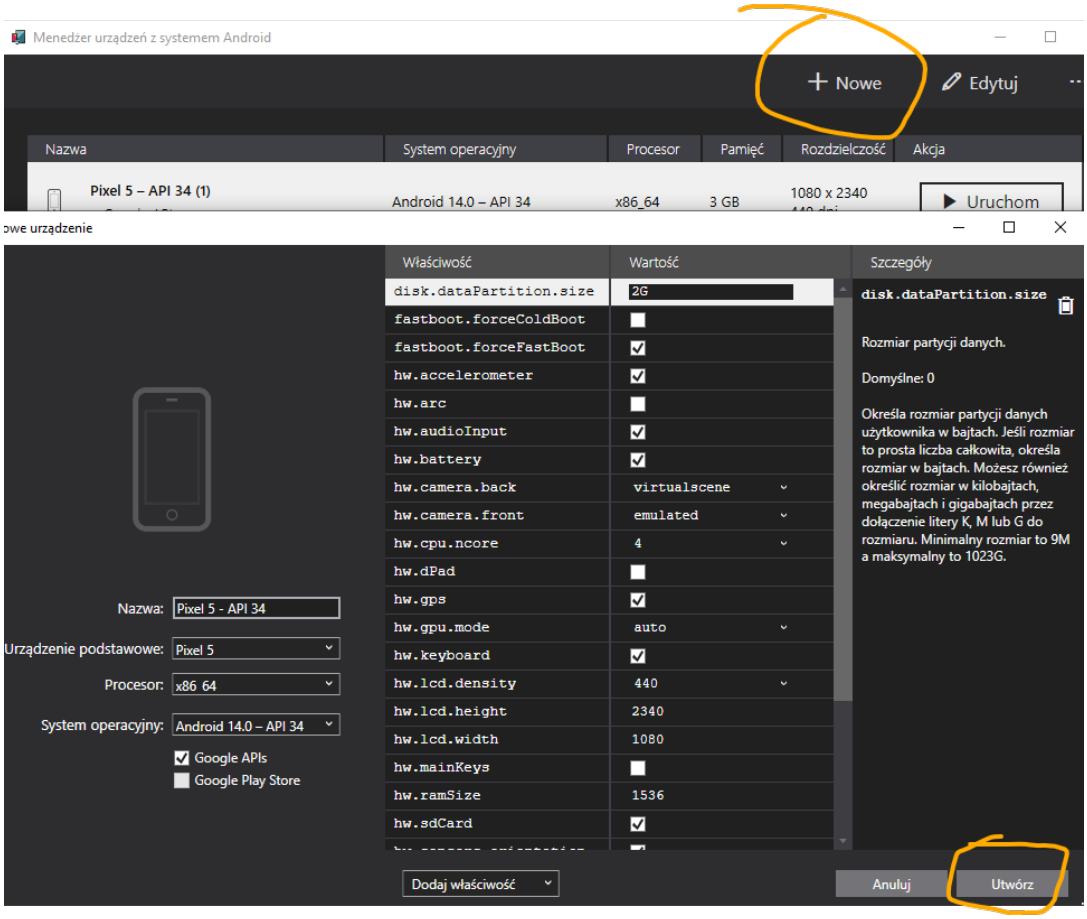




3) Następnie można włączyć projekt w VS i znaleźć opcję „Android Device Manager”. Można użyć do tego opcji „Search”:

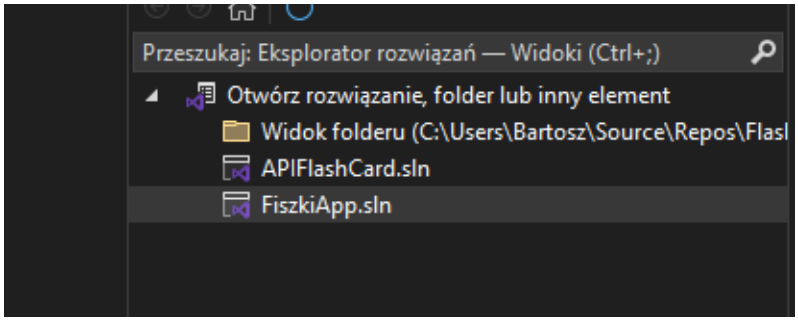


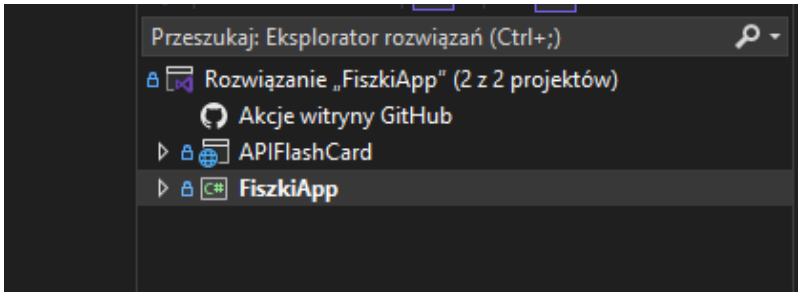
4) W otworzonym okienku wybrać „Nowe”, zaznaczyć np. urządzenie podstawowe: „Pixel 5” na systemie operacyjnym Android:



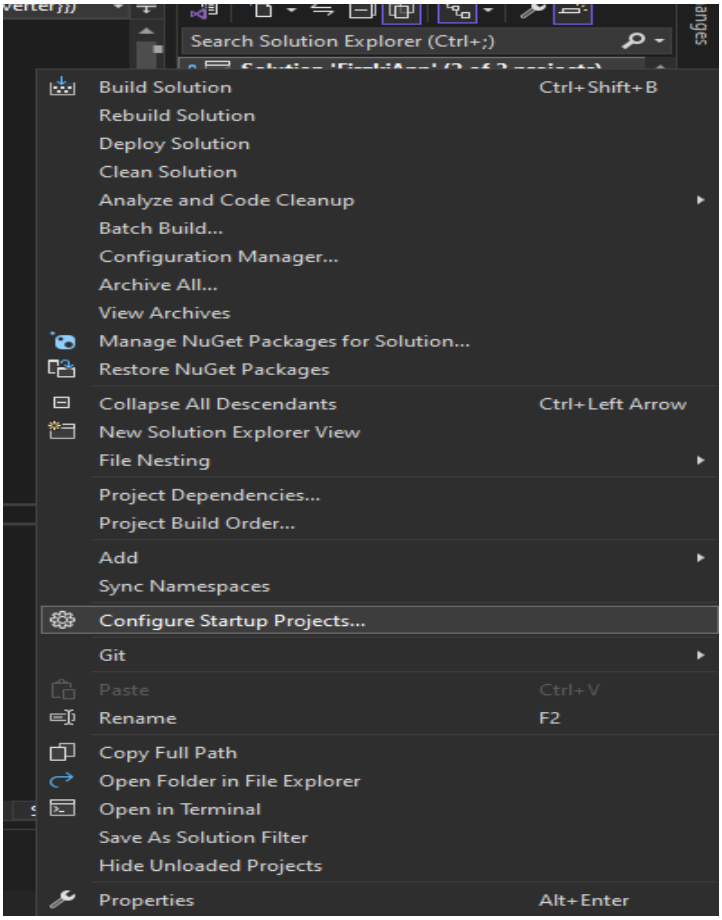
5) Klonujemy projekt z repozytorium

6) Uruchomi się tak, odczekujemy chwilę, aż wszystko się pobierze i klikamy dwukrotnie LPM na wybrane rozwiązanie by załadować projekt. Poniżej efekt przed i po

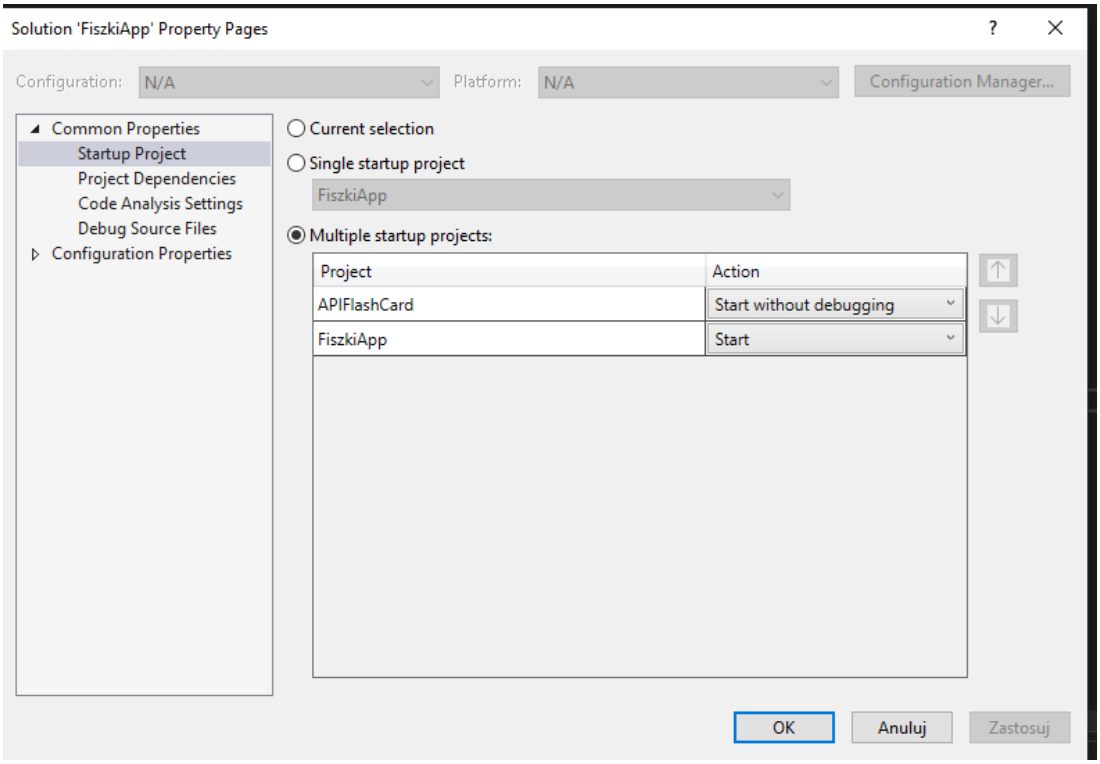




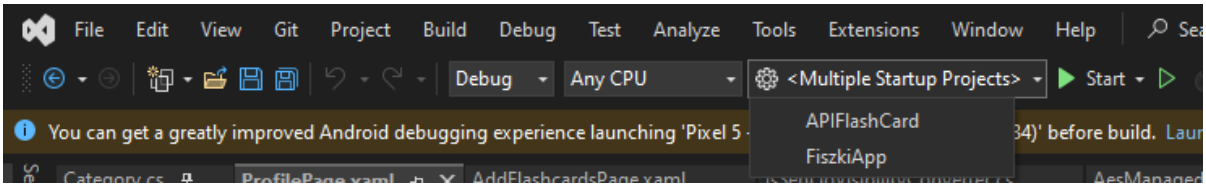
7) Na solucji kliknąć prawym przyciskiem myszy i wybrać opcje „Configure Startup Projects…”:



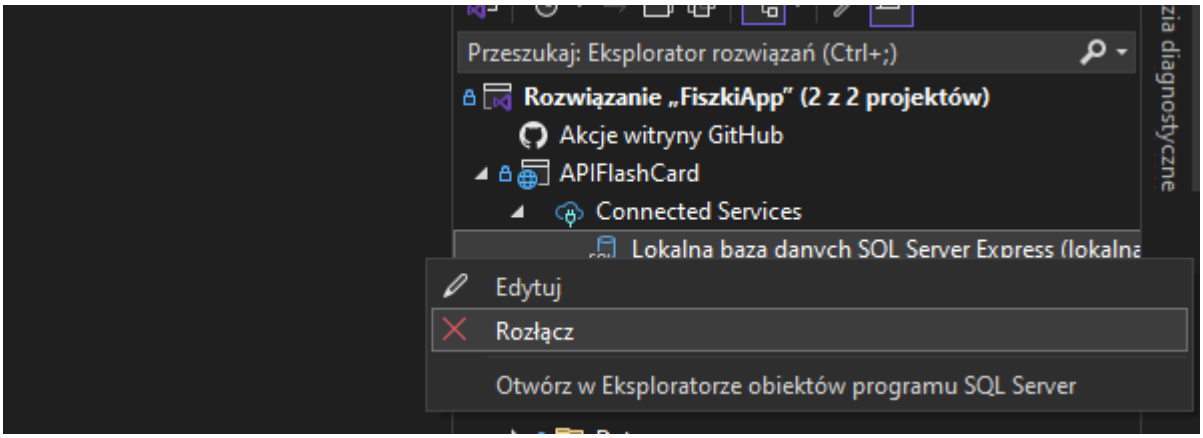
8) Wybrać opcję „Multiple startup projects”:



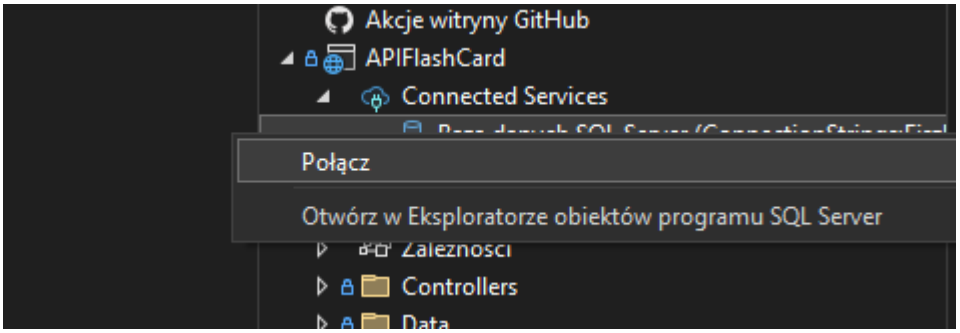
9) Upewnić się, że projekt jest ustawiony na <Multiple Startup Projects>:



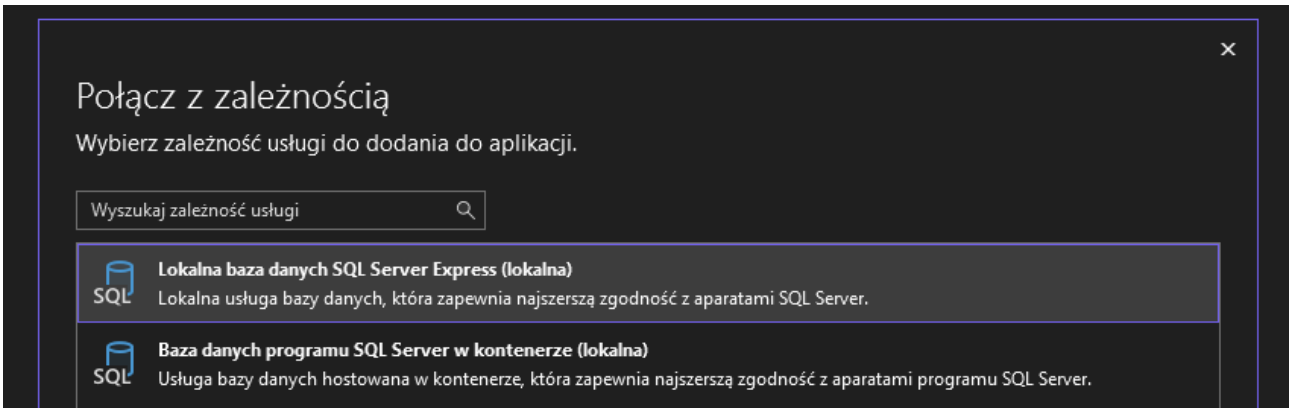
10) Rozłączyć istniejące połączenie z bazą danych SQL Server



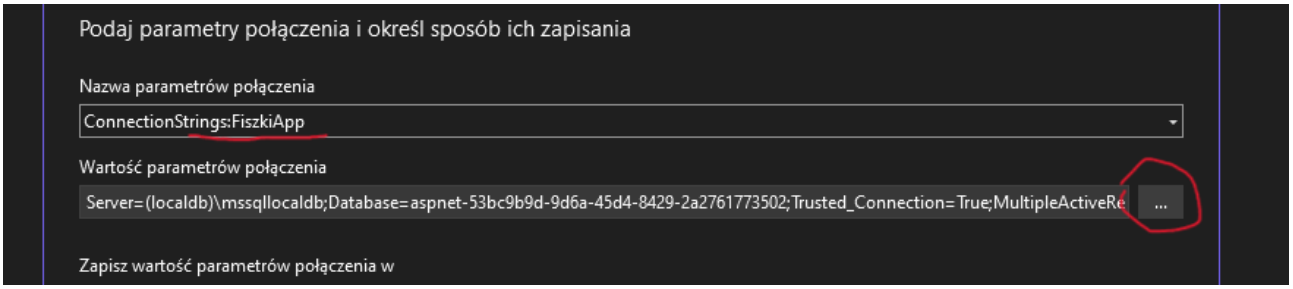
11) Kliknąć w „Połącz”



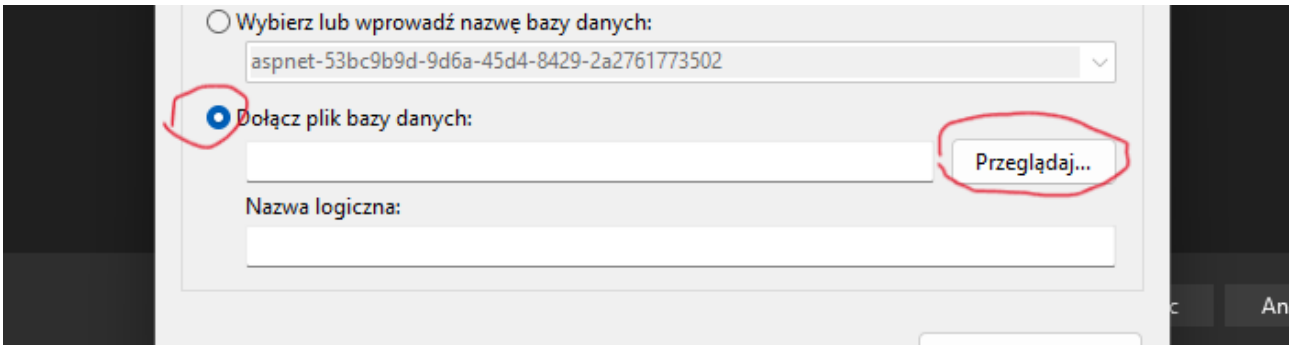
12) Wybrać Lokalna baza danych SQL Server Express (lokalna)

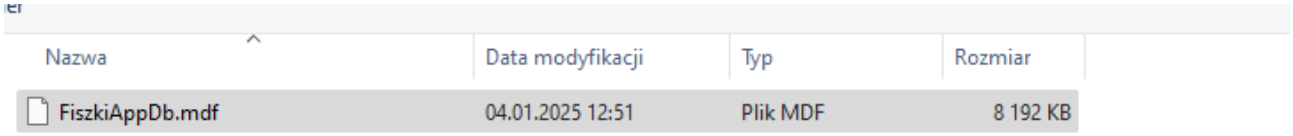


13) Wybrać używany w API ConnectionString:FiszkiApp, a następnie kliknąć w „ … ”

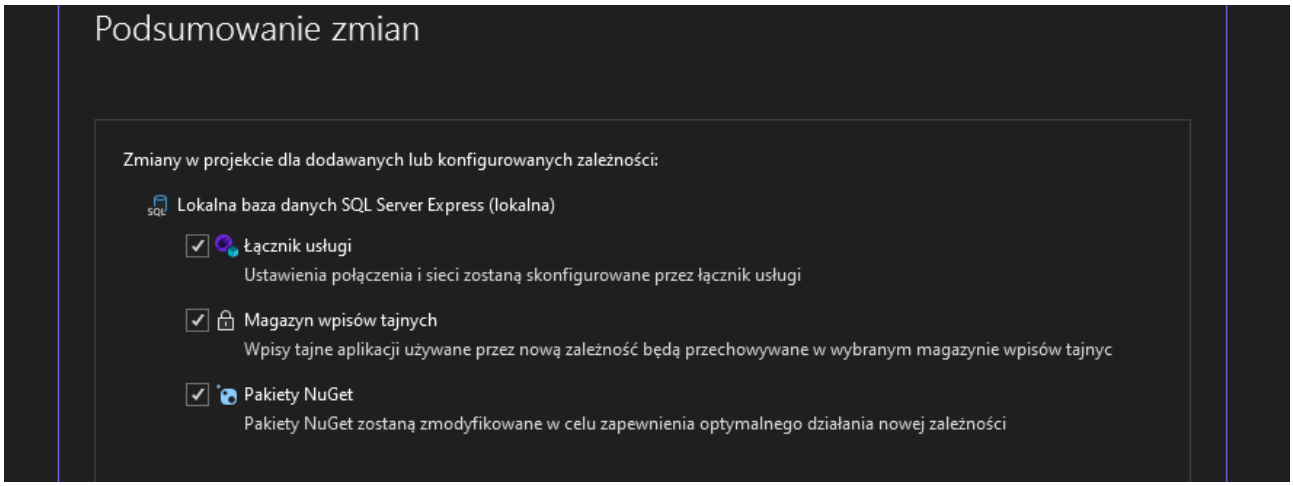


14) Pojawi nam się okno parametrów, gdzie na samym dole dodajemy plik bazy wskazując miejsce pobrania folderu Database z repozytorium projektu

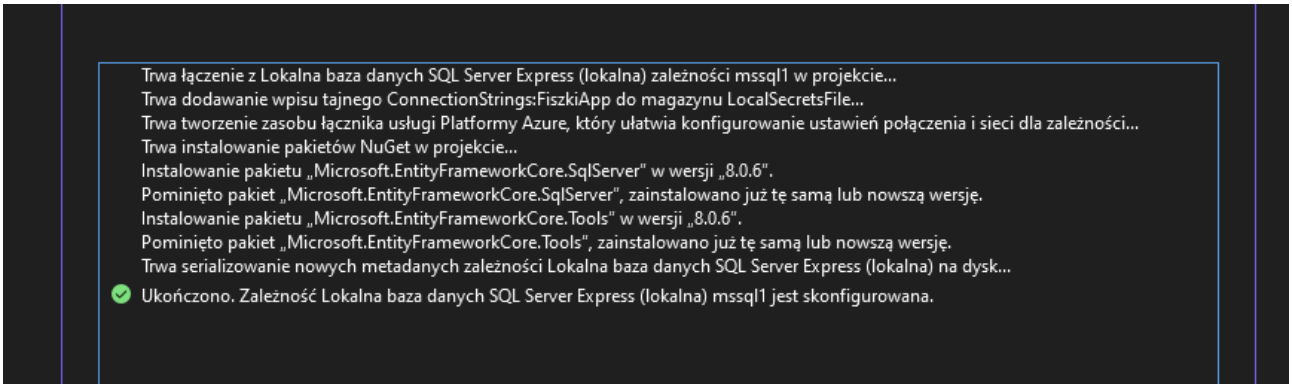




15) Klikamy dalej, tutaj pozostawiamy bez zmian



16) Efekt pomyślnego dodania bazy danych



17) Wszystko gotowe. Uruchamiamy projekt

6. Najciekawsze funkcjonalności:

- Klasa HttpClientService:

HttpClientService jest implementacją wzorca Singleton. Zapewnia jedną, współdzieloną instancję obiektu HttpClient do realizacji zapytań HTTP w całej aplikacji. Konstruktor klasy ustawia adres bazowy API oraz konfiguruje obsługę certyfikatów SSL (ignorując błędy w środowisku deweloperskim). Dostęp do instancji HttpClient uzyskuje się przez statyczną właściwość Instance. Dzięki temu rozwiązaniu zapewnia się wydajność i centralne zarządzanie połączeniami HTTP w aplikacji.

- System logowania w aplikacji:

System logowania wykorzystuje szyfrowanie AES do przechowywania haseł w formie zaszyfrowanej, zapewniając bezpieczeństwo danych. Klucze szyfrowania są przechowywane oddzielnie, co chroni hasła przed nieautoryzowanym dostępem.

Krótki opis klas i metod:

- **Klasa LogInQuery** zarządza procesem logowania użytkownika. Główna metoda, UserLogIn, wysyła zapytania HTTP w celu pobrania danych użytkownika i kluczy szyfrowania. Następnie hasło jest odszyfrowywane przy użyciu klasy AesManaged i porównywane z hasłem wprowadzonym przez użytkownika. Jeśli dane są poprawne, użytkownik zostaje zalogowany.

- **Klasa AuthService** zarządza stanem logowania. Metoda IsAuthenticatedAsync sprawdza, czy użytkownik jest zalogowany na podstawie zapisanych preferencji systemowych. Metoda Login inicjuje proces logowania poprzez wywołanie UserLogIn z LogInQuery, a po pomyślnym logowaniu zapisuje stan użytkownika w preferencjach. Metoda Logout usuwa dane logowania, wylogowując użytkownika.

- **Klasa AesManaged** odpowiada za szyfrowanie i deszyfrowanie danych za pomocą algorytmu AES. Metoda Encryption generuje klucz szyfrowania i wektor inicjujący (IV), aby zaszyfrować dane. Metoda Decryption odszyfrowuje dane przy użyciu podanego klucza i IV. W kontekście logowania, służy do odszyfrowania hasła przechowywanego w bazie danych.

- **Klasa UserRegistration** obsługuje proces rejestracji nowego użytkownika przypisując wszystko do User. Zawiera właściwości wymagane do rejestracji (np. imię, nazwisko, hasło) oraz metodę Validate, która sprawdza poprawność danych wejściowych (np. długość hasła, format e-maila). Po utworzeniu użytkownika dane User są czyszczone.

7. Kontrolery:

- **UserDetailsController**

- GetUserDetails(int userId): Pobiera szczegóły użytkownika na podstawie identyfikatora userId.
- CheckEmail(string email): Sprawdza, czy istnieje użytkownik z podanym adresem e-mail.
- PostUserDetails(UserDetails userDetails): Dodaje nowego użytkownika do bazy danych.

Nie zaimplementowano jeszcze usuwania użytkownika.

- **UserController**

- GetUserByUsername(string username): Pobiera użytkownika na podstawie nazwy użytkownika username.
- CheckUsername(string username): Sprawdza, czy istnieje użytkownik o podanej nazwie użytkownika.
- PostUser(User user): Dodaje nowego użytkownika do bazy danych.

Nie zaimplementowano jeszcze usuwania użytkownika oraz zmiany hasła.

- **EncryptionKeysController**

- GetEncryptionKeys(int userId): Pobiera klucze szyfrowania dla użytkownika na podstawie identyfikatora userId.
- PostEncryptionKeys(EncryptionKeys encryptionKeys): Dodaje nowe klucze szyfrowania dla użytkownika do bazy danych.

Nie zaimplementowano jeszcze usuwania użytkownika oraz zmiany hasła.

- CountriesController

- GetAllCountries(): Pobiera listę wszystkich krajów z bazy danych.
- GetCountryById(int id): Pobiera kraj na podstawie identyfikatora id.
- GetCountryByName(string countryName): Pobiera kraj na podstawie nazwy countryName.

- CategoryController

- AddCategory([FromBody] Category category): Dodaje nową kategorię do bazy danych.
- GetCategories(): Pobiera listę wszystkich kategorii z bazy danych.

- **FlashCardController**

- AddFlashCards(List<FlashCard> flashCards): Dodaje wiele fiszek do bazy danych w jednym żądaniu.
- GetFlashCardsByCategory(int categoryId): Pobiera fiszki należące do określonej kategorii, zwracając tylko pola FrontFlashCard i BackFlashCard.
- GetFlashCardsForDisplay(int categoryId, int page = 1): Pobiera fiszki z wybranej kategorii z podziałem na strony (po 10 fiszek na stronę), umożliwiając stronicowanie wyników.

- DynamicCategoryListController

SearchCategories(dynamic filters): Przeszukuje listę kategorii na podstawie dynamicznych filtrów, takich jak nazwa kategorii, nazwa użytkownika, poziom językowy i język użytkownika.

- FiltrujKategoriePoNazwie(string categoryName): Przeszukuje kategorie pasujące do podanej nazwy (częściowe dopasowanie).
- FiltrujKategoriePoUzytkowniku(string userName): Wyszukuje kategorie stworzone przez określonego użytkownika.
- FiltrujKategoriePoPoziomieJezykowym(string languageLevel): Pobiera kategorie o określonym poziomie znajomości języka.
- FiltrujKategoriePoJezyku(string userLanguage): Wyszukuje kategorie, gdzie podany język jest używany jako język frontowy lub tylny.

8. Modele:

Projekt zawiera 6 modeli:

- **Category:** posiada zestaw właściwości odpowiadających kolumnom w bazie danych, takich jak ID_Category (generowany automatycznie przez bazę danych), CategoryName, Country, czy LanguageLevel. Walidacja danych ogranicza się do

wymagań bazy danych, tj. długość string oraz obowiązkowość pól. UserID oraz ID_Country są kluczami obcymi, jednak modele w API nie deklarują bezpośrednio kluczy obcych. W związku z tym ustawiono je jako [Required], aby wymagać ich obecności w JSON przesyłanym z MAUI.

- **Countries:** model reprezentuje tabelę w bazie danych, zawierając właściwości takie jak ID_Country(generowany automatycznie przez bazę danych), Country, i Url. Walidacja ogranicza się do wymagań bazy danych, tj. obowiązkowości pola Country oraz maksymalnej długości string. Model nie posiada kluczy obcych.

- **EncryptionKeys:** ten model również odzwierciedla strukturę tabeli w bazie danych i zawiera właściwości takie jak ID_encryptionKeys (generowany automatycznie przez bazę danych), EncryptionKey, i IV. Walidacja obejmuje wymagania bazy danych, tj. obowiązkowość pola dla kluczy szyfrujących. ID_User jest kluczem obcym, jednak nie mamy go w modelu API pozostawiając tą właściwość tylko jako wymaganą, ponieważ zarządza tym warstwa aplikacji oraz baza danych.

- **User:** ten model definiuje właściwości odpowiadające kolumnom w bazie, takie jak ID_User (generowany automatycznie przez bazę danych), UserName, i UserPassword. Walidacja ogranicza się do wymagań bazy danych, np. obowiązkowości pól i długości UserName. Klucze obce nie są obecne w tym modelu. Wymagania JSON w aplikacji zapewniają integralność danych.

- **UserDetails:** posiada właściwości odpowiadające kolumnom w bazie danych, takie jak ID_Detailed, FirstName, LastName, Country, Email, i Avatar. Walidacja obejmuje wymagania bazy danych, np. maksymalną długość pól tekstowych oraz dopuszczenie wartości null (Avatar). ID_User jest kluczem obcym, lecz nie jest jawnie deklarowany w modelu API, co pozwala na elastyczność w przesyłaniu danych z aplikacji. Wymaganie klucza ID_User jest jednak wymagane na potrzeby pliku JSON przesyłanego z aplikacji MAUI.

- **FlashCard:** model reprezentuje tabelę w bazie danych, zawierając właściwości takie jak ID_flashcard(generowany automatycznie przez bazę danych), ID_category dla połączenia fiszek z kategorią oraz FrontFlashCard i BackFlashCard dla przechowywania przodu i tyłu fiszek. Walidacja ogranicza się do wymagań bazy danych, tj. obowiązkowości pól oraz maksymalnej długości string. Model nie posiada kluczy obcych.

9. Specyfikacja wykorzystanych technologii:

- **.NET 8 MAUI**

W aplikacji wykorzystano .NET MAUI, który umożliwia tworzenie aplikacji wieloplatformowych. Dzięki temu możliwe jest tworzenie aplikacji działającej na systemie Android przy użyciu jednego wspólnego kodu w C#. .NET MAUI zapewnia łatwe zarządzanie interfejsem użytkownika oraz dostępem do urządzeń mobilnych, co ułatwia rozwój aplikacji.

- C#
Język C# jest wykorzystywany zarówno w aplikacji mobilnej (MAUI), jak i w API. Służy do implementacji logiki biznesowej aplikacji oraz serwera, w tym obsługi autentykacji, zarządzania danymi oraz interakcji z bazą danych. C# umożliwia efektywne zarządzanie kodem zarówno po stronie klienta, jak i serwera.

- ASP.NET Core Web API
Do stworzenia interfejsu API zastosowano ASP.NET Core Web API, co pozwala na wygodne tworzenie i obsługę zasobów dostępnych przez HTTP. API zapewnia komunikację między aplikacją mobilną a bazą danych, umożliwiając operacje takie jak pobieranie danych użytkowników, zarządzanie kategoriami oraz kluczami szyfrowania.

- Entity Framework Core (EF Core)
W API użyto Entity Framework Core jako ORM (Object-Relational Mapping), co umożliwia mapowanie obiektów C# na tabele w bazie danych. EF Core wspiera operacje CRUD (tworzenie, odczyt, aktualizacja, usuwanie) na danych i zarządza relacjami między tabelami.

- SQLite
SQLite jest wykorzystywane w aplikacji mobilnej do przechowywania danych lokalnych na urządzeniu. Dzięki temu możliwe jest przechowywanie danych sesji, ustawień aplikacji oraz innych informacji, które mogą działać niezależnie od połączenia z internetem.

- HTTP/RESTful API
Komunikacja między aplikacją mobilną a API opiera się na protokole HTTP i standardzie RESTful. Aplikacja wysyła zapytania do API, aby pobierać dane lub wykonywać operacje na zasobach, takich jak użytkownicy, klucze szyfrowania czy kategorie.

- XAML
Do tworzenia interfejsu użytkownika aplikacji mobilnej użyto XAML, który umożliwia deklaratywne definiowanie widoków, layoutów i elementów UI. XAML pozwala na szybkie tworzenie estetycznych i funkcjonalnych interfejsów użytkownika.

- Json.NET (Newtonsoft.Json)
Json.NET jest wykorzystywane do serializacji i deserializacji danych w formacie JSON. Biblioteka ta umożliwia łatwą wymianę danych między aplikacją mobilną a API, co jest kluczowe w procesie komunikacji z serwerem.

- Git
Git jest systemem kontroli wersji wykorzystywanym w projekcie. Umożliwia zarządzanie kodem źródłowym, śledzenie zmian w projekcie i współpracę zespołową. Dzięki Git możliwe jest wprowadzanie zmian w projekcie w sposób kontrolowany i bezpieczny.

- MVVM (Model-View-ViewModel)
W aplikacji zastosowano architekturę MVVM, która pozwala na oddzielenie logiki aplikacji od interfejsu użytkownika. Model reprezentuje dane, View jest odpowiedzialne za interfejs, a ViewModel pośredniczy między nimi, obsługując logikę prezentacji i interakcji. Dzięki MVVM możliwe jest łatwiejsze testowanie, rozwój i utrzymanie aplikacji, a także łatwa współpraca z danymi i interfejsem użytkownika.