

AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Biblioteka

Autor:
Jakub Marek Tokarczyk
Mateusz Smaga

Prowadzący:
mgr inż. Dawid Kotlarski

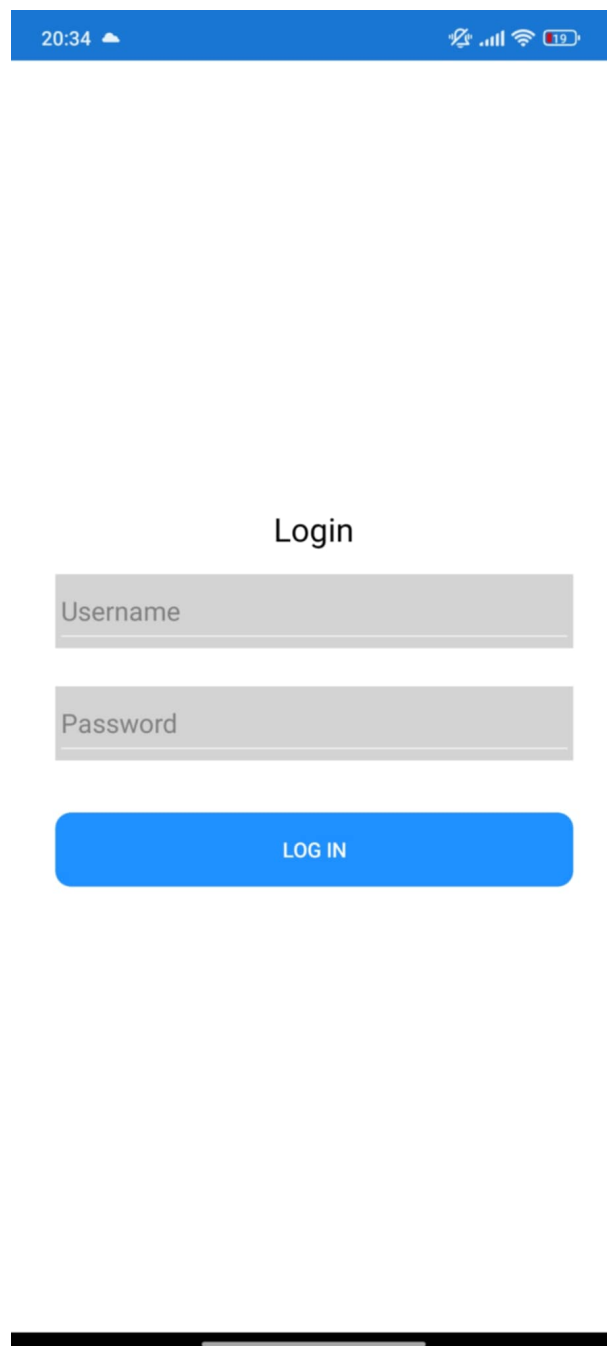
Nowy Sącz 2024

Spis treści

1. Ogólne określenie wymagań	3
2. Określenie wymagań szczegółowych	6
2.1. Zastosowane technologie do stworzenia aplikacji.	7
2.2. Przykładowy kod w Xamarin.	10
3. Projektowanie	21
3.1. Wykorzystane narzędzia:	21
3.1.1. Git	21
3.1.2. Github	21
3.1.3. Visual Studio 2019	21
3.1.4. C#	22
3.1.5. Firebase	22
3.2. Menu	22
3.3. Widoki	23
3.4. Pliki .xaml i .xaml.cs	24
3.5. Poszczególne widoki aplikacji	25
3.5.1. Utworzenie konta	25
4. Implementacja	30
5. Testowanie	34
6. Podręcznik użytkownika	35
Literatura	36
Spis rysunków	37
Spis tabel	38
Spis listingów	39

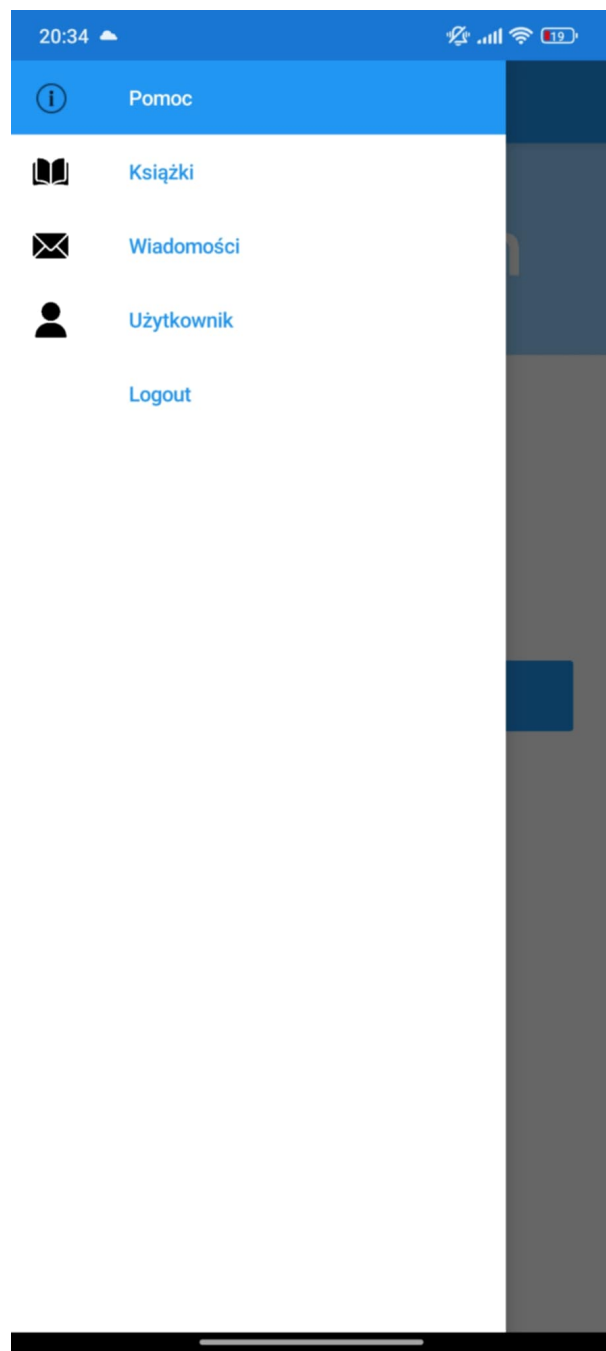
1. Ogólne określenie wymagań

Celem aplikacji Biblioteka jest dostarczenie użytkownikom tej aplikacji do efektywnego zarządzania własną kolekcją książek oraz umożliwienie interakcji z administratorem aplikacji. Aplikacja ma za zadanie uprościć proces kolekcjonowania książek poprzez funkcję skanowania, co pozwoli na szybsze wyszukiwanie książek w bazie danych. Dzięki temu, użytkownicy mogą szybko i wygodnie dodawać nowe pozycje do swojej biblioteki. Ponadto, aplikacja będzie posiadać opcje związane z edytowaniem oraz organizacją książek, co pozwoli na lepsze zarządzanie osobistymi ulubionymi zestawami książek. Aplikacja ma także za zadanie wspierać komunikację między hostem a użytkownikami. Host, pełniący rolę administratora aplikacji, będzie mógł przekazywać użytkownikom istotne informacje, takie jak wprowadzone aktualizacje, powiadomienia o zmianach, nowościach w aplikacji, zmianach w regulaminie lub inne komunikaty dotyczące samej obsługi aplikacji. Aplikacja Biblioteka skierowana jest do wszystkich osób, które posiadają własne kolekcje książek, niezależnie od ich wielkości. Może to być użyteczne narzędzie zarówno dla miłośników literatury jak i dla osób zajmujących się sprzedażą, wymianą, czy też wynajmem książek. Dzięki prostym funkcjom aplikacja pozwoli użytkownikom na organizację książek według własnych kryteriów (np. ulubione) a także na łatwe przeglądanie i zarządzanie zebranymi informacjami. Aplikacja będzie posiadała system logowania oraz rejestracji nowych użytkowników. Użytkownik, który nie posiada jeszcze konta, będzie mógł je utworzyć poprzez podanie podstawowych danych czyli nazwa użytkownika oraz hasło. Gdy użytkownik utworzy konto, będzie mógł zalogować się za pomocą nazwy użytkownika i hasła. Zakładka "Użytkownik" będzie zawierała szczegółowe informacje o aktualnie zalogowanym użytkowniku. W tej sekcji użytkownik będzie mógł przeglądać i edytować osobiste informacje takie jak: imię, nazwisko, nazwę użytkownika oraz adres e-mail. Zakładka "Książki" będzie pełniła funkcję zarządzania biblioteką książek użytkownika. Użytkownicy będą mieli możliwość dodania książek, ich edycje lub usunięcie. Zakładka "Wiadomości" pozwoli użytkownikom na przeglądanie otrzymanych wiadomości od administratora.



The image shows a mobile application login screen. At the top is a blue status bar with the time 20:34, a cloud icon, and icons for cellular signal, Wi-Fi, and a battery level of 19%. Below the status bar is a large white rectangular area. In the center of this area is the title "Login" in a bold, black, sans-serif font. Underneath the title are two stacked input fields with light gray backgrounds and thin gray borders. The first field is labeled "Username" and the second is labeled "Password". Below these fields is a blue rounded rectangular button with the text "LOG IN" in white, uppercase letters. At the very bottom of the screen is a black horizontal bar representing the home indicator, with a thin white line in the center.

Rys. 1.1. Logowanie



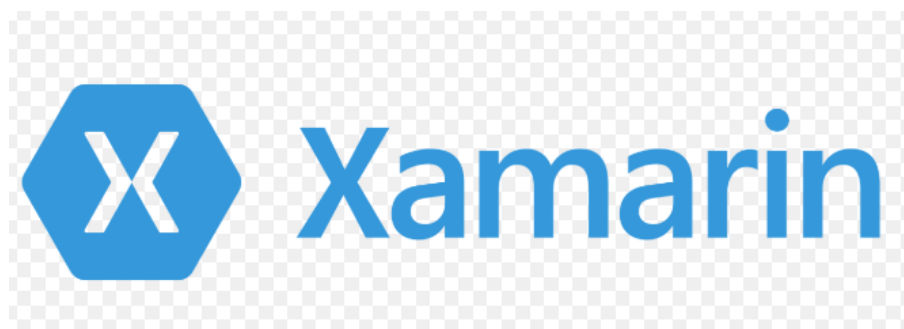
Rys. 1.2. Layout

2. Określenie wymagań szczegółowych

Aplikacja ma na celu ułatwienie użytkownikom gromadzenia, organizowania i zarządzania książkami w jednym miejscu, zarówno ulubionymi, starymi, jak i nowymi. Dzięki zastosowaniu nowoczesnych funkcji, takich jak dodawanie zdjęć książek, logowanie przez biometrię oraz automatyczne dostosowywanie interfejsu do jakości oświetlenia w danym pomieszczeniu czy otoczeniu, aplikacja zapewnia intuicyjne i wygodne narzędzie dla miłośników literatury. Użytkownicy będą mogli łatwo dodawać książki do swojej biblioteki, edytować dane książek oraz przeglądać je w zorganizowany sposób. Głównym założeniem jest, aby każdy użytkownik mógł przechowywać wszystkie swoje książki w jednym miejscu, niezależnie od tego, czy są to książki fizyczne, które już posiada, czy też tytuły, które planuje przeczytać. W aplikacji można dodać książki ręcznie, wpisując tytuł, autora, wydawnictwo i inne dane, ale również można wykorzystać aparat do dodania zdjęcia okładki danej książki. Po zalogowaniu do aplikacji użytkownik może przechodzić między różnymi funkcjami a logowanie może odbywać się za pomocą tradycyjnego hasła lub biometrii w przypadku administratora, czyli odcisku palca. Aplikacja automatycznie wykrywa warunki oświetleniowe otoczenia użytkownika. Jeśli w pomieszczeniu lub na zewnątrz jest jasno, aplikacja przechodzi w tryb dzienny, z jasnym interfejsem, który ułatwia przeglądanie książek i nawigację. Kiedy robi się ciemno, aplikacja automatycznie przełącza się w tryb nocny, zmieniając interfejs na ciemne kolory, co jest przyjemniejsze dla oczu i bardziej oszczędza baterię urządzenia. Użytkownik może także w każdej chwili ręcznie zmienić tryb, jeśli preferuje inny wygląd aplikacji. Dzięki integracji z aparatem, użytkownik może robić zdjęcia książek które dodaje do swojej biblioteki, lub robić zdjęcia swojej osoby i dodawać je jako zdjęcie profilowe. Każda z funkcji ma być prosta i intuicyjna a interfejs aplikacji będzie skoncentrowany na łatwości obsługi. Aplikacja będzie wykorzystywać bazę danych Firebase do przechowywania wszystkich danych użytkowników oraz ich książek. Firebase oferuje dynamiczne i bezpieczne przechowywanie danych, zapewniając przy tym synchronizację między różnymi urządzeniami. Oznacza to, że użytkownik będzie mógł korzystać z aplikacji na różnych urządzeniach, a jego dane zawsze będą aktualne, niezależnie od tego, z którego urządzenia korzysta. Środowiskiem programistycznym, w którym będzie tworzona aplikacja, jest Visual Studio 2019 z użyciem frameworka Xamarin. Zdecydowaliśmy się na to środowisko, ponieważ Visual Studio 2019 oferuje bogaty zestaw narzędzi deweloperskich oraz dobrą integrację z Xamarin, co pozwala na tworzenie aplikacji mobilnych działających na różnych systemach operacyjnych, takich jak Android

i iOS, z użyciem jednej bazy kodu. Xamarin umożliwia tworzenie aplikacji natywnych, co oznacza, że aplikacja będzie działać szybko i płynnie na obu platformach. Dodatkową zaletą jest wsparcie dla interfejsów mobilnych i łatwa integracja z różnymi funkcjami urządzeń, takimi jak aparat czy biometria. Zaletą Visual Studio 2019 jest jego rozbudowany interfejs, liczne wtyczki i narzędzia ułatwiające debugowanie oraz tworzenie aplikacji, a także wsparcie dla wielu języków programowania. Dzięki temu środowisku możemy szybko rozwijać aplikację a Xamarin zapewnia, że aplikacja będzie działać na wielu platformach bez konieczności pisania oddzielnych wersji na każdą z nich. Wadą może być to, że Visual Studio bywa zasobożerne, co sprawia, że może działać wolniej na słabszych komputerach. Xamarin natomiast, chociaż umożliwia pisanie aplikacji na różne platformy, nie jest tak elastyczny i wydajny jak natywne środowiska programistyczne dedykowane tylko dla jednej platformy.

2.1. Zastosowane technologie do stworzenia aplikacji.



Rys. 2.1. Logo Xamarin

Xamarin¹ to platforma open-source umożliwiająca tworzenie aplikacji mobilnych na różne systemy operacyjne, takie jak iOS, Android i Windows, przy użyciu jednego wspólnego kodu w języku C#. Xamarin jest częścią ekosystemu .NET i zapewnia narzędzia pozwalające na szybkie tworzenie aplikacji wieloplatformowych o natywnym wyglądzie i wydajności.

Najważniejsze cechy :

- *Wspólna baza kodu:* Xamarin umożliwia wykorzystanie nawet 90 procent wspólnego kodu przy tworzeniu aplikacji mobilnych na różne platformy. Oznacza to,

¹Xamarin[1]

że programiści mogą pisać jedną wersję logiki aplikacji (np. operacje na danych, integracje z API) w języku C#, a następnie wdrożyć ją na systemach iOS, Android, a także Windows, co pozwala obniżyć koszty i czas developmentu aplikacji,

- *Natywna wydajność:* Aplikacje stworzone w Xamarinie wykorzystują natywne interfejsy użytkownika i funkcje systemowe, co zapewnia wysoką wydajność,
- *Interfejs użytkownika:* Dwie opcje tworzenia UI:
 1. *Xamarin.Forms:* Tworzenie jednego, wspólnego interfejsu użytkownika dla wielu platform,
 2. *Xamarin.Native:* Tworzenie interfejsów indywidualnie dla każdej platformy,
- *Dostęp do API platformy:* Xamarin umożliwia korzystanie z pełnych funkcji dostępnych w systemach operacyjnych iOS i Android za pomocą kodu C#,
- *Integracja z Visual Studio:* Xamarin jest w pełni zintegrowany z Visual Studio, oferując wygodne środowisko pracy,
- *Biblioteki i komponenty:* Xamarin korzysta z ogromnego ekosystemu .NET, co oznacza, że aplikacje mogą wykorzystywać:
 1. *Gotowe biblioteki .NET:* do obsługi sieci, baz danych czy operacji na danych,
 2. *Komponenty NuGet:* dodatkowe pakiety, które rozszerzają funkcjonalność aplikacji,
 3. *Własne natywne komponenty:* programista może tworzyć własne komponenty specyficzne dla danej platformy,
- *Obsługa wielu platform jednocześnie:* Xamarin pozwala tworzyć aplikacje na:
 1. *iOS*
 2. *Android*
 3. *Windows*
 4. *MacOS*
- *Podsumowanie korzyści:*

1. *Szybsze dostarczanie aplikacji na wiele platform:* Dzięki jednej bazie kodu napisanej w języku C# programiści mogą jednocześnie tworzyć aplikacje na systemy iOS, Android i Windows, co znacząco skraca czas developmentu. Xamarin.Forms pozwala na tworzenie wspólnego interfejsu użytkownika, eliminując konieczność projektowania oddzielnych wersji dla każdej platformy,
2. *Obniżone koszty developmentu i utrzymania:* Użycie jednej bazy kodu zmniejsza zapotrzebowanie na zespoły specjalistyczne. Zamiast osobnych zespołów dla Androida i iOS, jeden zespół z doświadczeniem w .NET i C# może obsłużyć cały proces tworzenia aplikacji. Mniejsze nakłady pracy przy rozwoju i aktualizacjach oznaczają niższe koszty zarówno w fazie tworzenia, jak i późniejszego utrzymania aplikacji,
3. *Wydażność na poziomie aplikacji natywnych:* Xamarin kompiluje kod C# do natywnego kodu maszynowego, co pozwala uzyskać wydajność porównywalną z aplikacjami pisanymi w Swift dla iOS lub Kotlin dla Androida. Dostęp do natywnych API urządzeń zapewnia pełne wsparcie dla funkcji takich jak GPS, Bluetooth, aparat czy usługi płatności mobilnych.
4. *Rozbudowane narzędzia i wsparcie Visual Studio:* Xamarin jest w pełni zintegrowany z Visual Studio, co zapewnia dostęp do zaawansowanych narzędzi, takich jak debugowanie wieloplatformowe, projektowanie interfejsów za pomocą wizualnych edytorów oraz Hot Reload, który umożliwia szybkie wprowadzanie zmian w kodzie i ich natychmiastowe testowanie,
5. *Dostęp do wielu bibliotek:* Aplikacje tworzone w Xamarinie mogą wykorzystywać wszystkie zasoby dostępne w ekosystemie .NET, takie jak biblioteki do operacji sieciowych, obsługi baz danych czy zaawansowanej analizy danych. Programiści mają również dostęp do komponentów NuGet, co umożliwia łatwe dodanie gotowych rozwiązań do aplikacji. Xamarin wspiera także tworzenie własnych natywnych komponentów, które można wykorzystać w projektach wieloplatformowych, dostosowując je do specyficznych wymagań każdej platformy

Przykłady zastosowań Xamarina w aplikacjach mobilnych.

- *Aplikacje biznesowe:* Xamarin doskonale nadaje się do budowania aplikacji wspierających codzienną działalność firm. Dzięki dostępowi do natywnych funkcji urządzeń, takich jak GPS, kamera czy powiadomienia push, można tworzyć zaawansowane systemy ułatwiające zarządzanie przedsiębiorstwami,

- *Aplikacje konsumenckie:* Xamarin jest często wybierany do tworzenia aplikacji dla użytkowników indywidualnych, które muszą działać na różnych platformach i zapewniać atrakcyjny interfejs użytkownika oraz niezawodność np. aplikacje społecznościowe, aplikacje zakupowe lub aplikacje multimedialne,
- *Aplikacje edukacyjne i e-learningowe:* Xamarin jest świetnym wyborem dla aplikacji edukacyjnych, które często wymagają interaktywności, obsługi multimedialnych i możliwości działania na różnych urządzeniach,
- *Aplikacje IoT i wearables:* Xamarin pozwala na tworzenie aplikacji dla urządzeń IoT lub smartwatchy czy inteligentnych urządzeń domowych,
- *Gry mobilne i aplikacje rozrywkowe:* Xamarin pozwala na tworzenie gier i aplikacji rozrywkowych, które wykorzystują zaawansowane interfejsy graficzne i interakcje z użytkownikiem.

Xamarin znajduje zastosowanie wszędzie tam, gdzie wymagana jest multiplatformowość, wydajność oraz dostęp do natywnych funkcji urządzeń. To rozwiązanie sprawdza się zarówno w prostych aplikacjach użytkowych, jak i w zaawansowanych systemach wymagających integracji z różnymi technologiami.

2.2. Przykładowy kod w Xamarin.

```
1 using Xamarin.Forms;
2
3 namespace SimpleApp
4 {
5     public class MainPage : ContentPage
6     {
7         public MainPage()
8         {
9             Content = new Button
10             {
11                 Text = "Kliknij mnie!",
12                 Command = new Command(() => DisplayAlert("Witaj!",
13                 "Przycisk zosta Ć klikni Źty.", "OK"))
14             };
15         }
16 }
```

Listing 1. Przykładowy kod xamarin

Listing 1 odpowiada za:

- w linii dziewiątej tworzy przycisk jako główny element strony i ustawia jego właściwości
- w linii jedenastej tworzy tekst, który pojawia się na przycisku
- w linii dwunastej dodaje akcję, która zostanie wykonana po kliknięciu przycisku i wyświetla okienko z tytułem "Witaj!", oraz przyciskiem OK



Rys. 2.2. Visual Studio 2022

Visual Studio 2022² to zintegrowane środowisko programistyczne opracowane przez firmę Microsoft. Jest to jedno z najbardziej popularnych narzędzi do tworzenia aplikacji na różne platformy, w tym aplikacji mobilnych, webowych, desktopowych i wielu innych. Visual Studio 2022 wspiera wiele języków programowania, takich jak C#, C++, Python, JavaScript, TypeScript i inne.

Kluczowe cechy visual studio 2022:

- **Wieloplatformowość:** Visual Studio 2022 obsługuje wiele platform, takich jak Windows, macOS i Linux. Jest to środowisko do tworzenia aplikacji dla systemów Windows, Android, iOS, macOS, a także aplikacji webowych i chmurowych,
- **Zintegrowane narzędzia do debugowania:** Visual Studio 2022 oferuje rozbudowane narzędzia debugowania, które pozwalają na śledzenie kodu, analizowanie błędów w czasie rzeczywistym, a także monitorowanie wydajności

²Visual Studio 2022[2]

aplikacji. Debugowanie może odbywać się na różnych urządzeniach, w tym na emulatorach i rzeczywistych urządzeniach,

- **Rozbudowane funkcje IntelliSense:** IntelliSense to funkcja automatycznego uzupełniania kodu i sugestii dla programistów. Visual Studio 2022 zawiera inteligentne podpowiedzi, które pomagają w szybszym pisaniu kodu, poprawiając jego dokładność i redukując błędy,
- **Integracja z Git i systemami kontroli wersji:** Visual Studio 2022 wspiera pełną integrację z systemami kontroli wersji, takimi jak Git. Umożliwia to łatwe zarządzanie repozytoriami, synchronizowanie kodu z serwerem oraz współpracę z zespołami programistycznymi,
- **Ulepszony interfejs użytkownika:** Visual Studio 2022 oferuje nowoczesny i intuicyjny interfejs, który ułatwia programistom poruszanie się po projekcie. Dodatkowo, użytkownicy mogą dostosować układ i widoki zgodnie ze swoimi preferencjami,
- **Rozszerzenia i wtyczki:** Visual Studio 2022 obsługuje bogaty ekosystem wtyczek, które umożliwiają dodawanie nowych funkcji do środowiska IDE. Programiści mogą zainstalować wtyczki do pracy z bazami danych, chmurą, kontenerami Docker, a także różnymi frameworkami.

Integracja Visual Studio 2022 z Xamarin do tworzenia aplikacji mobilnych.

Xamarin jest frameworkiem pozwalającym na tworzenie aplikacji mobilnych na platformy Android, iOS i Windows z jednego wspólnego kodu źródłowego, wykorzystując język C# i .NET. Visual Studio 2022 zapewnia pełną integrację z Xamarin, oferując szereg narzędzi i funkcji, które sprawiają, że proces tworzenia aplikacji mobilnych staje się bardziej efektywny.

- *Instalacja Xamarin w Visual Studio 2022:* Aby korzystać z Xamarin w Visual Studio 2019, użytkownicy muszą zainstalować odpowiednią opcję podczas instalacji lub zaktualizować swoje IDE. W Visual Studio 2022 można łatwo dodać komponenty Xamarin, wybierając opcję „Mobile development with .NET” podczas instalacji,
- *Xamarin.Forms i Xamarin.Android w Visual Studio 2022:*

1. Xamarin.Forms to rozwiązanie umożliwiające tworzenie aplikacji mobilnych na Androida, iOS oraz Windows z użyciem jednej bazy kodu. Dzięki Xamarin.Forms możesz tworzyć aplikacje z graficznym interfejsem użytkownika (UI), który jest renderowany natywnie na każdej z platform,
 2. Xamarin.Android pozwala na tworzenie aplikacji dedykowanych tylko dla Androida, wykorzystując natywne API Androida,
- *Emulator Androida w Visual Studio 2022:* Visual Studio 2019 oferuje wbudowaną obsługę emulatorów Androida, co pozwala na testowanie aplikacji bez potrzeby korzystania z rzeczywistych urządzeń mobilnych. Można łatwo uruchamiać aplikacje na emulatorach Androida oraz testować aplikacje w różnych konfiguracjach urządzeń,
 - *Debugowanie aplikacji mobilnych:* Visual Studio 2022 oferuje zaawansowane narzędzia debugowania, które wspierają aplikacje stworzone przy użyciu Xamarin. Można debugować aplikację zarówno na emulatorach, jak i na rzeczywistych urządzeniach Android oraz iOS
 - *Testowanie na rzeczywistych urządzeniach:* Visual Studio 2022 umożliwia również testowanie aplikacji Xamarin na rzeczywistych urządzeniach Android i iOS. Można podłączyć urządzenie mobilne bezpośrednio do komputera, uruchomić aplikację na nim i debugować ją w czasie rzeczywistym,
 - *Wsparcie dla bibliotek i paczek NuGet:* Xamarin w Visual Studio 2022 wspiera pakiety NuGet, co pozwala na łatwe dodawanie zewnętrznych bibliotek i frameworków do aplikacji mobilnych. Może to obejmować biblioteki do integracji z bazami danych, obsługi powiadomień push, komunikacji sieciowej i innych funkcji mobilnych.

Visual Studio 2022 to potężne narzędzie, które oferuje pełną integrację z Xamarin, umożliwiając tworzenie aplikacji mobilnych na Androida, iOS i inne platformy z wykorzystaniem jednego kodu źródłowego. Dzięki temu środowisku deweloperzy mogą korzystać z zaawansowanych narzędzi debugowania, testowania i emulatorów, co znacząco przyspiesza proces tworzenia i wdrażania aplikacji mobilnych. Integracja z Xamarin umożliwia tworzenie zarówno aplikacji wieloplatformowych jak i dedykowanych aplikacji natywnych.



Rys. 2.3. Xamarin kontra Kotlin

Kotlin³ to nowoczesny, wieloplatformowy, statycznie typowany język programowania, który został opracowany przez firmę JetBrains. Język ten został zaprojektowany, aby zapewnić większą produktywność, bezpieczeństwo oraz łatwość pisania kodu w porównaniu do Javy, z którą jest w pełni interoperacyjny. Od 2017 roku Kotlin jest oficjalnie wspieranym językiem do tworzenia aplikacji na platformę Android.

- *Język programowania:*

1. *Xamarin:* Xamarin używa języka C#, który jest częścią ekosystemu .NET. C# jest popularnym językiem programowania, znanym ze swojej czytelnej składni, silnego wsparcia dla obiektowego podejścia oraz bogatej biblioteki klas, pozwala na szerokie wykorzystanie istniejącego kodu .NET
2. *Kotlin:* Kotlin jest nowoczesnym językiem programowania stworzonym przez firmę JetBrains, który działa na maszynie wirtualnej Javy. Kotlin jest pełnoprawnym językiem, który Google wybrało jako preferowany język do tworzenia aplikacji na Androida, jest interoperacyjny z Javą, co pozwala na płynne przechodzenie z Javy na Kotlin w projektach Androidowych

- *Wieloplatformowość:*

³Kotlin[3]

1. *Xamarin*: Xamarin umożliwia tworzenie wieloplatformowych aplikacji mobilnych z jednym wspólnym kodem źródłowym. Dzięki Xamarin.Forms można tworzyć interfejs użytkownika, który jest renderowany natywnie na różnych platformach, co pozwala na zaoszczędzenie czasu i zasobów,
 2. *Kotlin*: Kotlin, z natury, jest skierowany głównie na platformę Android. Istnieje projekt Kotlin Multiplatform, który pozwala na współdzielenie kodu między Androidem, iOS i innymi platformami, ale jest to nadal młodsza technologia w porównaniu do Xamarin, kotlin Multiplatform nie oferuje pełnej integracji z interfejsem użytkownika na wielu platformach, co oznacza, że trzeba korzystać z osobnych kodów dla UI dla każdej platformy,
- *Wydajność*:
 1. *Xamarin*: Xamarin kompiluje kod do natywnego kodu maszynowego dla każdej platformy. Dzięki temu aplikacje Xamarin działają niemalże z taką samą wydajnością jak aplikacje natywne napisane w Javie,
 2. *Kotlin*: Kotlin jest w pełni kompatybilny z Javą, a aplikacje Kotlinowe na Androidzie są kompilowane do tego samego bytecode'u co Java. Wydajność aplikacji Kotlin w porównaniu do aplikacji napisanych w Javie jest bardzo podobna, ponieważ są one oparte na tej samej maszynie wirtualnej,
 - *Ekosystem i wsparcie*:
 1. *Xamarin*: Xamarin jest częścią większego ekosystemu .NET, co daje dostęp do szerokiej biblioteki i narzędzi takich jak ASP.NET, Entity Framework, Azure i wiele innych. To sprawia, że jest to doskonały wybór dla programistów już korzystających z tego ekosystemu,
 2. *Kotlin*: Kotlin jest intensywnie wspierany przez Google w kontekście aplikacji Android. Oferuje również wsparcie dla różnych narzędzi, takich jak Kotlin/Native do tworzenia aplikacji dla iOS, macOS, Linuxa, itp. Kotlin jest stale rozwijany przez JetBrains i społeczność open-source, jednak jego wsparcie nie jest tak szerokie, jak w przypadku Xamarin w kontekście innych platform poza Androidem,
 - *Narzędzia i IDE*:
 1. *Xamarin*: Xamarin jest ściśle zintegrowany z Visual Studio, co zapewnia bogate środowisko programistyczne, które obejmuje debugowanie, testo-

wanie, CI/CD i wsparcie dla natywnych API. Visual Studio oferuje pełne wsparcie dla Xamarin i jest jednym z najlepszych IDE do pracy nad aplikacjami mobilnymi,

2. *Kotlin*: Kotlin jest wspierany przez Android Studio, które jest dedykowanym środowiskiem programistycznym dla Androida. Android Studio oferuje rozbudowane narzędzia do budowania, debugowania i testowania aplikacji Kotlinowych.

Dlaczego wybraliśmy środowisko xamarin?

Po pierwsze, Xamarin oferuje wieloplatformowość, umożliwiając tworzenie aplikacji mobilnych na Androida, iOS i Windows z jednym wspólnym kodem źródłowym. Dzięki temu oszczędza się czas i zasoby, ponieważ nie ma potrzeby pisania oddzielnego kodu dla każdej platformy. Z kolei Kotlin, mimo że ma projekt Kotlin Multiplatform, jest głównie skoncentrowany na Androidzie i nie oferuje pełnej wieloplatformowości, szczególnie w zakresie interfejsów użytkownika. Po drugie, Xamarin jest częścią ekosystemu .NET, co umożliwia łatwą integrację z innymi narzędziami i bibliotekami tego ekosystemu, takimi jak ASP.NET, Azure, Entity Framework czy C#. Dzięki temu programiści, którzy już pracują w tym ekosystemie, mogą łatwo wykorzystać swoje dotychczasowe zasoby i przyspieszyć proces tworzenia aplikacji. Kotlin koncentruje się głównie na Androidzie, co ogranicza dostęp do podobnych narzędzi i bibliotek w kontekście innych platform.

Dodatkowo, Xamarin oferuje pełny dostęp do natywnych API dla platform Androida i iOS, co daje większą elastyczność i możliwość wykorzystania zaawansowanych funkcji. Kotlin, mimo że współpracuje z Javą i może korzystać z natywnych API Androida, nie oferuje natywnego wsparcia dla iOS, co sprawia, że tworzenie aplikacji na obie platformy w jednym projekcie może być bardziej skomplikowane. Xamarin korzysta także z Visual Studio, które jest jednym z najbardziej zaawansowanych IDE, oferującym szeroką gamę narzędzi do debugowania, testowania i wdrażania aplikacji. Kotlin działa głównie w ekosystemie Android Studio, które jest doskonałe dla aplikacji Androidowych, ale nie oferuje wsparcia dla innych platform.

Ponadto Xamarin używa C#, który jest jednym z najpopularniejszych języków programowania, szczególnie w środowisku enterprise. Dla zespołów, które już pracują z C# i .NET, Xamarin jest naturalnym wyborem, pozwalającym na wykorzystanie istniejącego kodu i zasobów w nowych aplikacjach mobilnych. Kotlin jest świetnym wyborem dla programistów Androida, ale nie oferuje tej samej uniwersalności, szczególnie w ekosystemie .NET



Rys. 2.4. Github

GitHub⁴ to platforma internetowa oparta na systemie kontroli wersji Git, która umożliwia programistom przechowywanie, zarządzanie i współpracę nad kodem źródłowym w ramach projektów oprogramowania. GitHub jest szczególnie popularny wśród zespołów deweloperskich i społeczności open-source, ponieważ oferuje szeroką gamę funkcji do wspólnej pracy nad kodem.

Korzyści z korzystania z GitHub podczas tworzenia aplikacji mobilnej w grupie:

- *Współpraca w zespole:* GitHub umożliwia zespołom programistów efektywną współpracę nad jednym projektem. Każdy członek zespołu może pracować nad swoją częścią aplikacji, wprowadzać zmiany w oddzielnych gałęziach (branchach) i łatwo łączyć je z główną wersją kodu po przeglądzie. Dzięki temu unika się konfliktów, a zespół może równocześnie rozwijać różne elementy aplikacji,
- *Kontrola wersji:* GitHub działa na bazie systemu Git, który pozwala na ścisłą kontrolę wersji kodu. Dzięki temu każdy członek zespołu może wprowadzać zmiany w kodzie, a cała historia zmian jest zapisywana. Jeśli coś pójdzie nie tak, można łatwo cofnąć się do poprzednich wersji kodu, co daje dużą pewność i bezpieczeństwo pracy nad aplikacją,
- *Kodowanie w trybie równoległym:* GitHub pozwala na pracę nad tymi samymi funkcjami przez wielu deweloperów jednocześnie, bez ryzyka nadpisania czy utraty kodu. Przez wykorzystanie branchy, każdy programista może rozwijać

⁴GitHub[4]

swoją część aplikacji w oddzielnym wątku, a później połączyć swoją pracę z główną gałęzią kodu. Funkcja pull request umożliwia przegląd kodu przed jego połączeniem, co zapewnia jakość kodu i pozwala na dyskusję nad rozwiązaniami,

- *Wersjonowanie kodu i śledzenie błędów:* GitHub umożliwia efektywne śledzenie błędów i zadań przy pomocy funkcji issues. Programiści mogą zgłaszać błędy, propozycje nowych funkcji i zadania do wykonania, a następnie przypisywać je do konkretnych członków zespołu. Dzięki temu łatwo jest śledzić postęp prac, priorytetyzować zadania i zarządzać projektem w sposób uporządkowany,
- *Bezpieczeństwo i dostępność:* GitHub zapewnia, że cały kod jest bezpiecznie przechowywany w chmurze. W przypadku pracy nad aplikacją mobilną w grupie, GitHub umożliwia ustawienie różnych poziomów dostępu do kodu: publicznego lub prywatnego. Dzięki temu można kontrolować, kto ma dostęp do wrażliwego kodu aplikacji, zapewniając odpowiedni poziom bezpieczeństwa,
- *Dokumentacja i zarządzanie projektem:* GitHub umożliwia dodawanie plików dokumentacji w repozytorium, dzięki czemu cała dokumentacja dotycząca aplikacji, jak i jej procesów deweloperskich, może być przechowywana obok kodu.
- *Spółeczność i wsparcie:* GitHub ma ogromną społeczność programistów, co ułatwia znalezienie wsparcia w razie problemów, czy nawet inspiracji do nowych rozwiązań. Istnieje również możliwość korzystania z gotowych bibliotek i frameworków, które można łatwo zaadoptować do swojego projektu, co przyspiesza rozwój aplikacji.



Rys. 2.5. Firebase

Firebase⁵ to platforma opracowana przez Google, która oferuje zestaw narzędzi i usług do tworzenia aplikacji mobilnych i webowych. Firebase zapewnia różnorodne funkcjonalności, które pozwalają na łatwe zarządzanie danymi, autentykację użytkowników, powiadomienia push, analitykę oraz przechowywanie plików, a wszystko to z wbudowanym wsparciem dla skalowalności i wydajności. Dzięki tym usługom Firebase jest popularnym wyborem dla twórców aplikacji, którzy chcą skupić się na rozwoju produktu bez konieczności zarządzania infrastrukturą serwerową.

Zalety bazy danych Firebase:

- *Baza danych w czasie rzeczywistym:* Firebase oferuje Firebase Realtime Database i Firestore, które pozwalają na przechowywanie i synchronizowanie danych w czasie rzeczywistym. Dzięki temu zmiany w danych są natychmiastowo odzwierciedlane na wszystkich urządzeniach użytkowników, co jest szczególnie przydatne w aplikacjach wymagających interakcji w czasie rzeczywistym, jak np. czaty,

⁵Firebase[5]

- *Autentykacja użytkowników:* Firebase zapewnia wbudowaną usługę Firebase Authentication, która obsługuje logowanie za pomocą różnych metod, takich jak e-mail, hasło, logowanie przez Google, Facebook, Twitter czy inne zewnętrzne systemy. Dzięki tej usłudze proces logowania i rejestracji użytkowników jest znacznie uproszczony, a bezpieczeństwo jest na wysokim poziomie,
- *Przechowywanie plików Firebase Storage:* umożliwia przechowywanie i zarządzanie plikami, takimi jak obrazy, wideo czy dokumenty, w sposób skalowalny i bezpieczny. Przechowywanie plików w chmurze pozwala na łatwy dostęp z urządzeń mobilnych, a także zapewnia automatyczne skalowanie w zależności od potrzeb aplikacji,
- *Skalowalność i wydajność:* Firebase jest platformą opartą na chmurze, która automatycznie skalowalna, co oznacza, że nie trzeba się martwić o zarządzanie serwerami czy infrastrukturą. Usługi takie jak Firebase Hosting czy Firestore zapewniają wysoką wydajność aplikacji, nawet gdy liczba użytkowników gwałtownie rośnie,
- *Łatwe dodawanie funkcji backendowych:* Firebase zapewnia gotowe usługi backendowe, takie jak Firebase Authentication, Cloud Firestore, Realtime Database, Firebase Storage i Firebase Cloud Messaging, które można łatwo zaimplementować w aplikacji Xamarin. Integracja z tymi usługami nie wymaga zarządzania serwerami, co pozwala zaoszczędzić czas i zasoby,

Integracja Firebase z Xamarin daje wiele korzyści, takich jak możliwość łatwego tworzenia aplikacji mobilnych na wiele platform, wykorzystanie zaawansowanych funkcji backendowych bez potrzeby zarządzania serwerami oraz możliwość łatwego dodawania powiadomień push, autentykacji użytkowników, przechowywania danych w czasie rzeczywistym i monitorowania aplikacji. Firebase w połączeniu z Xamarinem sprawia, że tworzenie aplikacji mobilnych jest szybsze, bardziej wydajne i skalowalne.

3. Projektowanie

3.1. Wykorzystane narzędzia:

3.1.1. Git

Git⁶ to system kontroli wersji, który pozwala programistom śledzić zmiany w kodzie źródłowym w czasie rzeczywistym, co umożliwia łatwe zarządzanie historią projektu. Git umożliwia tworzenie różnych gałęzi (branchy), na których programiści mogą pracować nad różnymi funkcjonalnościami lub poprawkami bez ingerencji w główną wersję kodu. System ten wspiera również współpracę między członkami zespołu, ponieważ pozwala na integrację zmian wprowadzone przez różnych deweloperów. Git pozwala na szybkie cofanie zmian, co daje bezpieczeństwo i elastyczność w pracy nad projektem.

3.1.2. Github

GitHub to platforma oparta na systemie Git, która umożliwia hostowanie repozytoriów kodu w chmurze. GitHub nie tylko zapewnia zdalne przechowywanie projektów, ale także oferuje zaawansowane narzędzia do współpracy, takie jak pull requesty, które umożliwiają przeglądanie i zatwierdzanie zmian przed ich włączeniem do głównej wersji kodu. Platforma oferuje również funkcje zarządzania projektami, takie jak issues do śledzenia błędów i zadań, a także wikis i documentation, które pomagają w tworzeniu pełnej dokumentacji projektu. GitHub wspiera również integrację z narzędziami CI/CD, co pozwala na automatyzację procesów budowania i testowania aplikacji.

3.1.3. Visual Studio 2019

Visual Studio 2022 to zintegrowane środowisko programistyczne stworzone przez firmę Microsoft, które umożliwia programistom tworzenie aplikacji na różne platformy, w tym Windows, macOS, aplikacje webowe, mobilne i gry. Visual Studio 2019 obsługuje szeroki zakres języków programowania, takich jak C#, C++, Python, JavaScript, a także technologie i frameworki takie jak .NET, Xamarin, ASP.NET, Unity, oraz wiele innych. Jest to jedno z najpotężniejszych i najbardziej zaawansowanych środowisk programistycznych, które łączy w sobie edytor kodu, narzędzia do debugowania, integrację z systemami kontroli wersji, oraz funkcje do testowania i deployowania aplikacji.

⁶Strona główna systemu kontroli wersji Git[6]

3.1.4. C#

C#⁷ to obiektowy język programowania stworzony przez Microsoft, który jest szeroko stosowany w ramach platformy .NET. C# jest językiem o wysokiej wydajności, charakteryzującym się czytelną składnią, co czyni go przyjaznym dla programistów. Dzięki silnemu typowaniu i wbudowanemu zarządzaniu pamięcią (dzięki środowisku uruchomieniowemu .NET), C# jest odpowiedni do tworzenia aplikacji desktopowych, mobilnych, webowych, a także gier. C# jest wykorzystywany w takich technologiach jak ASP.NET do tworzenia aplikacji webowych, Xamarin do aplikacji mobilnych oraz Unity do tworzenia gier. Jego elastyczność i wsparcie dla wielu paradygmatów programowania, w tym programowania obiektowego, asynchronicznego i funkcyjnego, sprawiają, że jest to jeden z najpopularniejszych języków programowania.

3.1.5. Firebase

Firebase⁸ to platforma chmurowa od Google, zaprojektowana z myślą o tworzeniu aplikacji mobilnych i webowych. Firebase oferuje szereg usług, które znacznie upraszczają rozwój aplikacji backendowych, takich jak Firebase Realtime Database i Cloud Firestore do przechowywania danych w czasie rzeczywistym, Firebase Authentication do zarządzania logowaniem i rejestracją użytkowników oraz Firebase Cloud Messaging do wysyłania powiadomień push. Dodatkowo Firebase oferuje Firebase Analytics, które umożliwia śledzenie aktywności użytkowników i optymalizację aplikacji na podstawie zebranych danych. Platforma zapewnia także łatwą integrację z innymi usługami Google, takimi jak Google Ads, oraz umożliwia automatyzację procesów testowania i wdrażania aplikacji. Firebase jest szczególnie cenione za to, że pozwala na szybki rozwój aplikacji bez potrzeby zarządzania własną infrastrukturą serwerową, co oszczędza czas i zasoby.

3.2. Menu

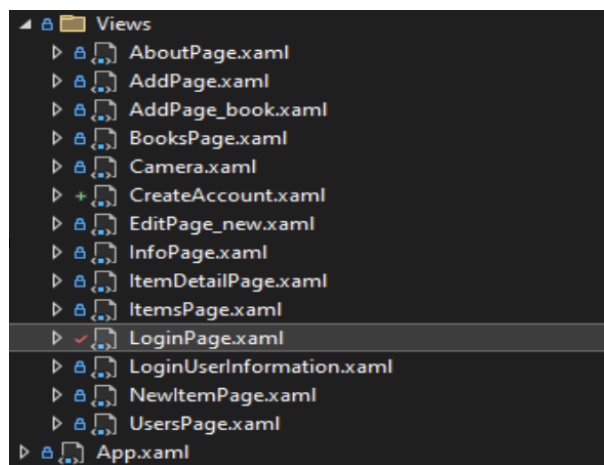
Menu wysuwane posiada następujące zakładki: Książki jest to główna sekcja aplikacji, prowadząca do zbioru zapisanych książek. Po kliknięciu w ikonkę przeniesie nas, do zbioru naszych dotychczasowych książek, gdzie będzie można je dodawać i usuwać. Wiadomości to sekcja z wiadomościami lub uaktualnieniami aplikacji od hosta do użytkowników aplikacji, po wejściu w sekcję, będziemy mogli zobaczyć wia-

⁷C#[7]

⁸Firebase[5]

domości od hosta dotyczące na przykład aktualizacji aplikacji. Użytkownik to sekcja profilu użytkownika, gdzie po wejściu w nią, można zobaczyć dane na swój temat takie jak imię, nazwisko, e-mail i również je edytować. Logout to opcja, która umożliwia wylogowanie się z aplikacji, co zapewni prywatność i bezpieczeństwo dostępu, po kliknięciu zostaniemy przekierowani do ekranu logowania do aplikacji. Każda z ikon oraz tekst w menu są w jasnym niebieskim kolorze, co pasuje do ogólnego motywu kolorystycznego aplikacji i zapewnia spójność wizualną. Samo tło menu jest białe, co daje kontrast w stosunku do ciemnego motywu głównego formularza i ułatwia czytelność opcji nawigacyjnych.

3.3. Widoki



Rys. 3.1. Widoki

Widoki, możemy zobaczyć je na rysunku 3.1 reprezentują interfejs użytkownika. W tym przypadku, odpowiedzialne są za:

AddPage.xml - Widok umożliwiający dodanie nowego elementu np. wpisu, zawiera pola do wprowadzania danych,

AddPagebook.xml - jest dedykowany dodawaniu książek, umożliwia wprowadzanie szczegółów książki, takich jak tytuł, autor,

BooksPage.xml - zawiera listę książek, umożliwia interakcje z książkami takie jak usuwanie, edytowanie,

Camera.xml - jest to widok integrujący funkcję aparatu, który pozwala użytkownikowi na robienie zdjęć,

CreateAccount.xml - jest to Widok rejestracji nowego użytkownika, zawiera pola do wprowadzenia danych,

EditPage.xml - Widok umożliwiający edycję danych istniejącego elementu, po-

zwala na ich modyfikację oraz zapisanie zmian,

InfoPage.xaml - strona wyświetlająca szczegółowe informacje o konkretnym elemencie, może być używana np. do pokazania pełnych informacji o książce czy użytkowniku,

ItemDetailPage.xaml - może służyć do wyświetlania szczegółowych danych o konkretnym wpisie, podobnie jak InfoPage.xaml,

ItemsPage.xaml - widok wyświetlający listę elementów, np. użytkowników czy książek, zawiera listę z możliwością interakcji np. kliknięcie w element

LoginPage.xaml - strona logowania do aplikacji, zawiera pola na login i hasło oraz przycisk logowania. Oferuje także opcje utworzenia konta,

LoginUserInformation.xaml - widok wyświetlający informacje o zalogowanym użytkowniku, pokazując dane takie jak imię i nazwisko użytkownika, e-mail,

NewItemPage.xaml - widok umożliwiający dodanie nowego elementu,

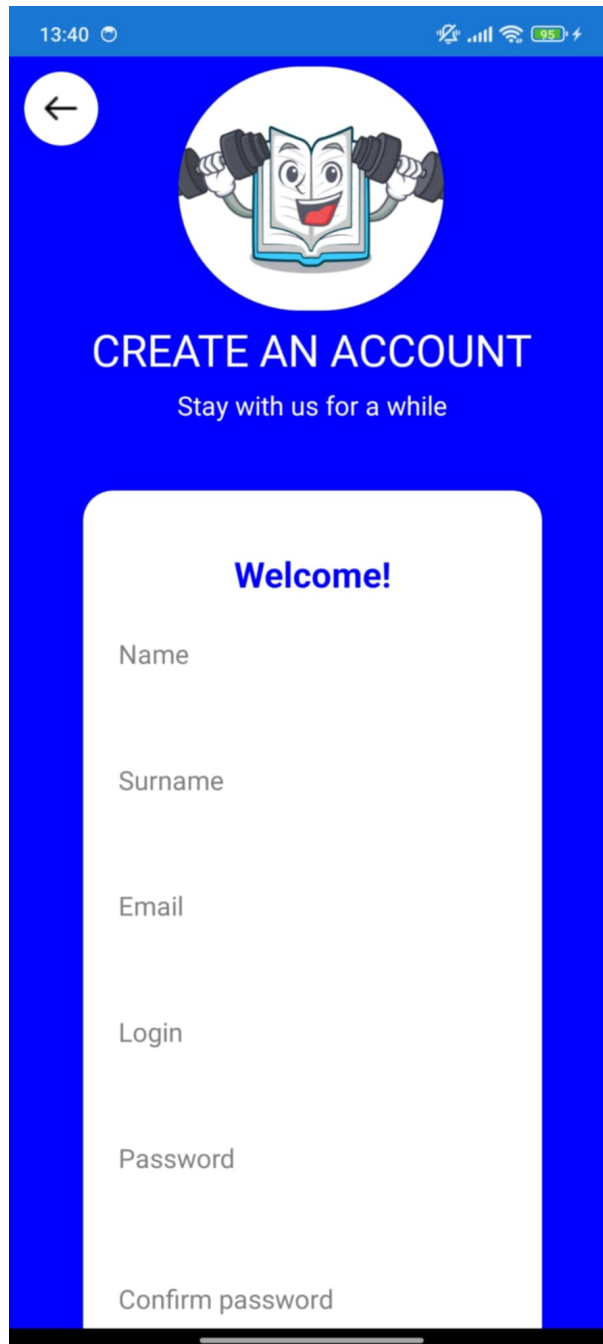
UsersPage.xaml - widok listy użytkowników.

3.4. Pliki .xaml i .xaml.cs

Plik .xaml to specjalny typ pliku używany w technologiach takich jak WPF, Xamarin.Forms czy MAUI, który służy do definiowania interfejsu użytkownika w sposób deklaratywny. Umożliwia to tworzenie widoków graficznych poprzez określanie układu elementów interfejsu, takich jak przyciski, pola tekstowe, siatki, obrazy, etykiety czy inne kontrolki wizualne. Dzięki składni XML, pliki .xaml są przejrzyste i łatwe do edycji zarówno dla programistów, jak i projektantów. Można w nich precyzyjnie opisać, jakie elementy interfejsu mają się znajdować na danej stronie, jakie mają mieć style, rozmiary czy rozmieszczenie. Z kolei plik .cs, czyli tak zwany kod-behind jest bezpośrednio powiązany z plikiem .xaml i zawiera logikę aplikacji, która jest związana z daną stroną lub widokiem. Każdy plik .xaml ma odpowiadający mu plik .xaml.cs, gdzie znajdują się definicje metod i zdarzeń, takich jak reakcje na kliknięcia przycisków czy wprowadzanie danych przez użytkownika. Na przykład, jeśli mamy stronę LoginPage.xaml, która wyświetla formularz logowania, jej kod-behind będzie znajdować się w pliku LoginPage.xaml.cs. W tym miejscu można zaimplementować zachowanie tej strony, np. walidację danych logowania, przejście do innej strony po udanym logowaniu lub wyświetlenie komunikatu o błędzie. Dzięki podziałowi na .xaml i .cs, programiści mogą oddzielić część wizualną od logiki biznesowej, co czyni kod bardziej przejrzystym i łatwiejszym w utrzymaniu. Takie podejście wspiera również pracę zespołową, ponieważ różne osoby mogą jednocześnie zajmować się projektowaniem interfejsu i implementowaniem jego funkcjonalności.

3.5. Poszczególne widoki aplikacji

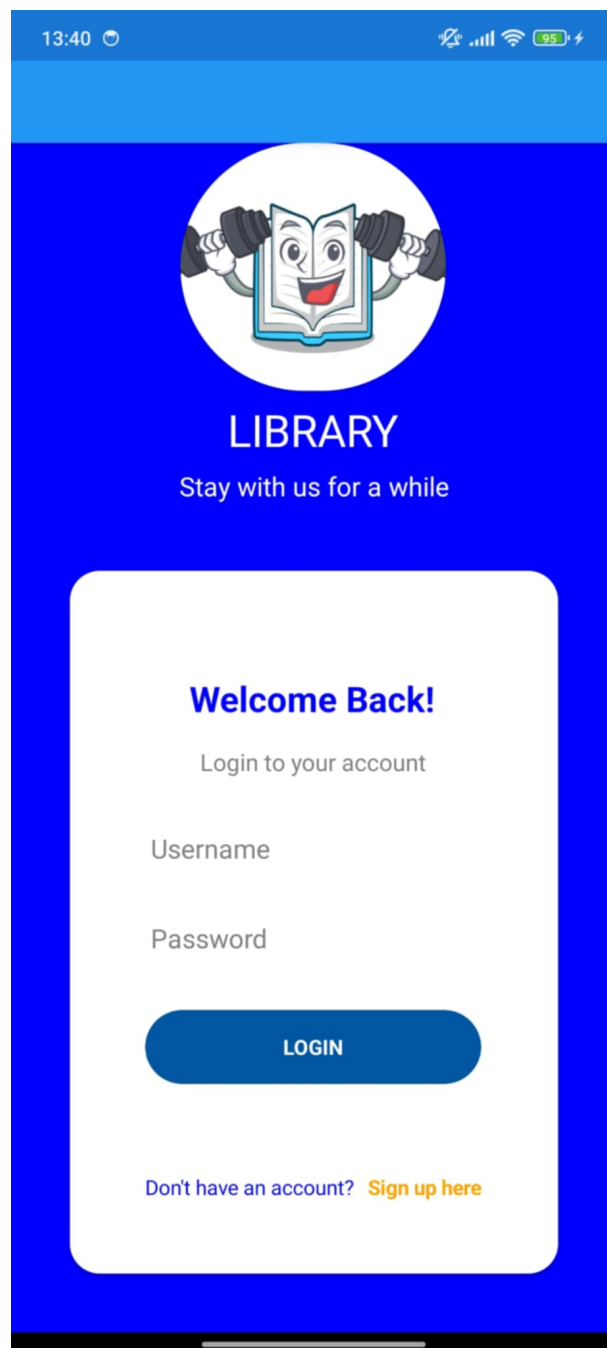
3.5.1. Utworzenie konta



Rys. 3.2. Utworzenie konta

Widok tworzenia konta jest zaprezentowany na rysunku 3.2. Proces rejestracji wymaga od użytkownika wprowadzenia kilku istotnych informacji. Po pierwsze, użytkownik musi podać swoje imię oraz nazwisko. Kolejnym krokiem jest podanie adresu e-mail. Użytkownik będzie musiał również wymyślić unikalny login, który

będzie używany do logowania się do swojego konta, oraz hasło. Hasło należy wpisać dwukrotnie, aby potwierdzić, że użytkownik nie popełnił błędu podczas jego tworzenia. Wszystkie te dane są niezbędne do założenia konta i umożliwiają użytkownikowi dostęp do aplikacji po zakończeniu procesu rejestracji.



Rys. 3.3. Logowanie

Użytkownik będzie miał możliwość zalogowania się do aplikacji tak jak jest to pokazane na rysunku 3.3, korzystając z wcześniej utworzonego loginu oraz hasła. Jeśli jednak użytkownik nie ma jeszcze konta, będzie mógł je założyć, klikając przycisk

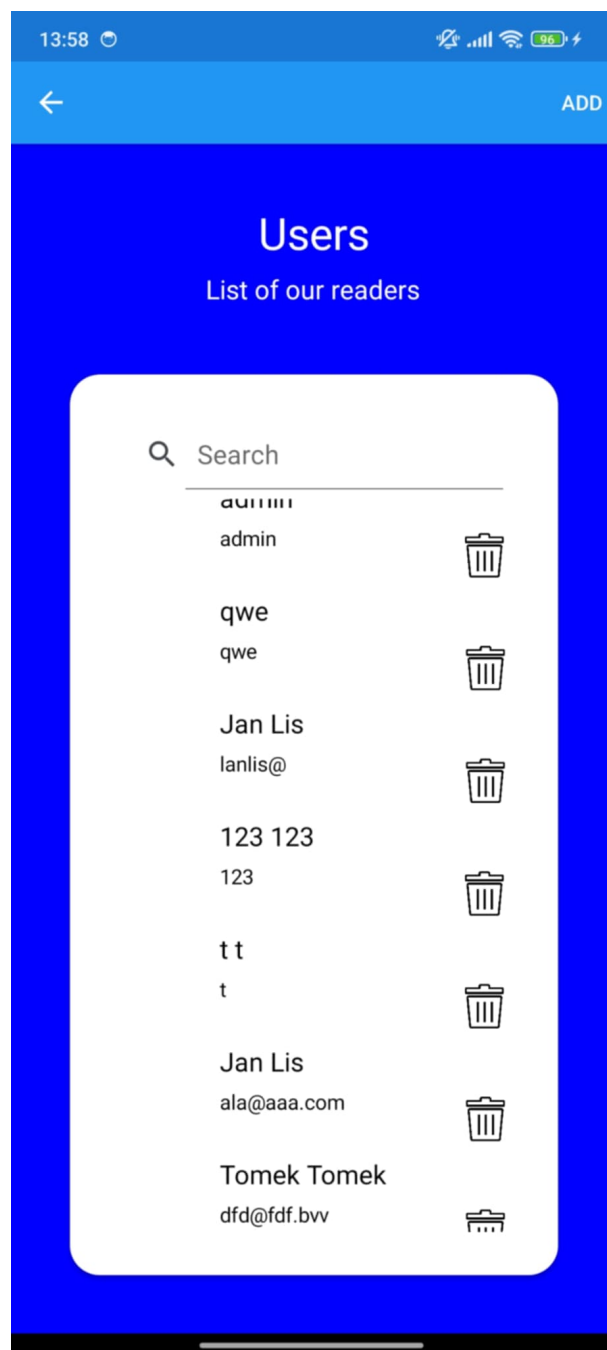
”Sign up here”. Po kliknięciu w ten link, użytkownik zostanie przekierowany do formularza rejestracyjnego, gdzie będzie musiał podać swoje dane, takie jak imię, nazwisko, adres e-mail, login oraz hasło. Dzięki temu procesowi, użytkownik zyska dostęp do aplikacji, a jego konto zostanie utworzone, umożliwiając logowanie się w przyszłości.



Rys. 3.4. Edycja książek

Ekran edytowania książek ukazany na rysunku 3.4 umożliwia użytkownikowi zarządzanie swoimi książkami w aplikacji. Na tym ekranie użytkownik ma możliwość

dodawania nowych książek do swojej biblioteki, edytowania już istniejących oraz usuwania książek, których już nie chce mieć w swojej kolekcji. Użytkownik może również wprowadzić nazwę książki oraz jej opis, co pozwala na lepsze zarządzanie zawartością biblioteki. Dodatkowo, ekran oferuje funkcję wyszukiwania, dzięki której użytkownik może szybko odnaleźć książkę po jej nazwie. To ułatwia zarządzanie dużą liczbą książek, umożliwiając szybkie dotarcie do poszukiwanej pozycji.



Rys. 3.5. Użytkownicy

Administrator aplikacji będzie miał pełen dostęp do listy wszystkich użytkowników,

tak jak możemy to ujrzyć na rysunku 3.5, którzy zarejestrowali się w aplikacji. Na ekranie administracyjnym będą widoczne dane użytkowników, takie jak ich imię, nazwisko oraz adres e-mail, który został podany podczas procesu rejestracji. Administrator będzie miał również możliwość wyszukiwania użytkowników na podstawie ich imienia i nazwiska, co umożliwi szybkie odnalezienie konkretnej osoby w przypadku dużej liczby użytkowników. Ponadto, administrator będzie miał uprawnienia do usuwania użytkowników z listy, co oznacza, że będzie mógł usunąć konto wybranego użytkownika, jeśli zajdzie taka potrzeba np. w przypadku naruszenia regulaminu aplikacji.

4. Implementacja

```

1 <FlyoutItem Title="About" Icon="icon_about.png">
2     <ShellContent Route="AboutPage" ContentTemplate="{DataTemplate
    local:AboutPage}" />
3 </FlyoutItem>
4 <FlyoutItem Title="Browse" Icon="icon_feed.png">
5     <ShellContent Route="ItemsPage" ContentTemplate="{DataTemplate
    local:ItemsPage}" />
6 </FlyoutItem>
7 <FlyoutItem Title="Users" Icon="">
8     <ShellContent Route="UsersPage" ContentTemplate="{DataTemplate
    local:UsersPage}" />
9 </FlyoutItem>
10
11 <MenuItem Text="Logout" StyleClass="MenuItemLayoutStyle" Clicked="
    OnMenuItemClicked">
12 </MenuItem>

```

Listing 2. Zakładki

- **Linijka 1** Deklaracja pozycji menu w nawigacji bocznej z tytułem "About" i przypisaną ikoną,
- **Linijka 2** Deklaracja zawartości powiązanej z pozycją menu "About". Określa trasę AboutPage oraz stronę AboutPage jako widok ładowany po wybraniu tej pozycji,
- **Linijka 4** Deklaracja pozycji menu w nawigacji bocznej z tytułem "Browse" i przypisaną ikoną,
- **Linijka 5** Deklaracja zawartości powiązanej z pozycją menu "Browse". Określa trasę ItemsPage oraz stronę ItemsPage jako widok ładowany po wybraniu tej pozycji,
- **Linijka 7** Deklaracja pozycji menu w nawigacji bocznej z tytułem "Users",
- **Linijka 8** Deklaracja zawartości powiązanej z pozycją menu "Users". Określa trasę UsersPage oraz stronę UsersPage jako widok ładowany po wybraniu tej pozycji
- **Linijka 11** Deklaracja pozycji menu typu MenuItem z tekstem "Logout", po kliknięciu wywoływana jest metoda OnMenuItemClicked do wylogowania użytkownika.

```
1 FirebaseClient firebaseClient = new FirebaseClient("https://
    bibliotekapro-eeddd-default-rtdb.firebaseio.com/");
2
3 //add
4 public async Task<bool> Save(User user)
5 {
6     var data = await firebaseClient.Child(nameof(User)).PostAsync(
        JsonConvert.SerializeObject(user));
7     if(!string.IsNullOrEmpty(data.Key))
8     {
9         return true;
10    }
11    return false;
12 }
13 //save
14 public async Task<List<User>> GetAll()
15 {
16     return (await firebaseClient.Child(nameof(User)).OnceAsync<User
        >()).Select(item => new User
17     {
18         Email = item.Object.Email,
19         Name = item.Object.Name,
20         Image = item.Object.Name,
21         Id = item.Key
22     }).ToList();
23 }
24 //get id
25 public async Task<User>GetById(string id)
26 {
27     return (await firebaseClient.Child(nameof(User) + "/" + id).
        OnceSingleAsync<User>());
28 }
29 //update
30 public async Task<bool> Update(User user)
31 {
32     await firebaseClient.Child(nameof(User) + "/" + user.Id).
        PutAsync(JsonConvert.SerializeObject(user));
33     return true;
34 }
35 //delete
36 public async Task<bool>Delete(string id)
37 {
38     await firebaseClient.Child(nameof(User) + "/" + id).DeleteAsync
        ();
39     return true;
```

40 }

Listing 3. Metody

- **Linijka 1** Inicjalizacja klienta Firebase, który łączy się z bazą danych w podanym URL,
- **Linijki 4-12** Zapisuje nowego użytkownika do Firebase pod kluczem User, konwertując obiekt użytkownika do formatu JSON, sprawdza, czy Firebase zwróciło klucz dla dodanego rekordu, co potwierdza powodzenie operacji,
- **Linijki 14-23** Pobiera wszystkie obiekty użytkowników z bazy danych i mapuje je na listę obiektów User, dane są przetwarzane tak, aby zawierały klucz użytkownika jako Id Email, Name oraz Image,
- **Linijki 25-28** Pobiera pojedynczego użytkownika z Firebase na podstawie jego identyfikatora, wykorzystuje pełną ścieżkę do użytkownika, aby zwrócić dane,
- **Linijki 30-34** Wysyła zaktualizowane dane użytkownika do bazy Firebase, korzystając z identyfikatora Id, nadpisuje istniejące dane użytkownika w bazie na podstawie podanego identyfikatora,
- **Linijki 36-40** Usuwa użytkownika z Firebase, korzystając z jego identyfikatora do odnalezienia odpowiedniego rekordu w bazie, dane użytkownika są trwale usuwane z bazy danych, co jest potwierdzone przez zwrócenie true.

```

1  public class User: IUserWithId
2  {
3      public string Id { get; set; }
4      public string Name { get; set; }
5      public string Email { get; set; }
6      public string Image { get; set; }
7
8      public string Login { get; set; }
9      public string Password { get; set; }
10 }
```

Listing 4. Obiekt User

- **Linijki 1-10** Klasa User reprezentuje użytkownika w systemie, zawierając właściwości takie jak Id, Name, Email, Image, Login oraz Password. Każda z

tych właściwości jest publiczna, co oznacza, że może być dostępna i modyfikowana. Właściwość Id jest unikalnym identyfikatorem użytkownika, a pozostałe właściwości przechowują dane osobowe i logowania użytkownika.

```
1 public interface IUserWithId
2 {
3     string Id { get; set; }
4 }
```

Listing 5. Interfejs IUserWithId

- **Linijki 1-4** Interfejs IUserWithId definiuje właściwość Id, celem tego jest zapewnienie, że wszystkie klasy implementujące ten interfejs mają pole Id, które identyfikuje obiekt. Interfejs IUserWithId: Gwarantuje, że każdy obiekt typu User będzie miał właściwość Id.

```
1 public class Book: IUserWithId
2 {
3     public string Id { get; set; }
4     public string Name { get; set; }
5     public string Author { get; set; }
6     public string Image { get; set; }
7 }
```

Listing 6. Klasa book

- **Linijki 1-7** Klasa Book implementuje interfejs IUserWithId, co oznacza, że każdy obiekt Book musi mieć unikalny identyfikator, zawierając właściwości takie jak Id, Name, Author, i Image. Pozostałe właściwości zawierają dane, które umożliwiają przechowywanie i wyświetlanie szczegółów książki, takie jak tytuł, autor i obrazek okładki.

5. Testowanie

6. Podręcznik użytkownika

Bibliografia

- [1] *xamarin*. URL: <https://www.cdata.com/kb/articles/xamarin.rst> (term. wiz. 03.12.2024).
- [2] *Visual Studio 2022*. URL: <https://visualstudio.microsoft.com/pl/vs/getting-started/> (term. wiz. 10.12.2024).
- [3] *Kotlin*. URL: <https://kotlinlang.org/> (term. wiz. 03.12.2024).
- [4] *Github*. URL: <https://github.com/> (term. wiz. 03.12.2024).
- [5] *Firebase*. URL: <https://firebase.google.com/> (term. wiz. 03.12.2024).
- [6] *Git*. URL: <https://git-scm.com/> (term. wiz. 09.11.2024).
- [7] *c#*. URL: <https://www.w3schools.com/cs/index.php> (term. wiz. 03.12.2024).

Spis rysunków

1.1. Logowanie	4
1.2. Layout	5
2.1. Logo Xamarin	7
2.2. Visual Studio 2022	11
2.3. Xamarin kontra Kotlin	14
2.4. Github	17
2.5. Firebase	19
3.1. Widoki	23
3.2. Utworzenie konta	25
3.3. Logowanie	26
3.4. Edycja książek	27
3.5. Użytkownicy	28

Spis tabel

Spis listingów

1.	Przykładowy kod xamarin	10
2.	Zakładki	30
3.	Metody	31
4.	Obiekt User	32
5.	Interfejs IUserWithId	33
6.	Klasa book	33