

Delphi i sztuczna inteligencja: randka w ciemno czy miłość od pierwszego wejrzenia?



Zdzisław Sroczyński
23.03.2023

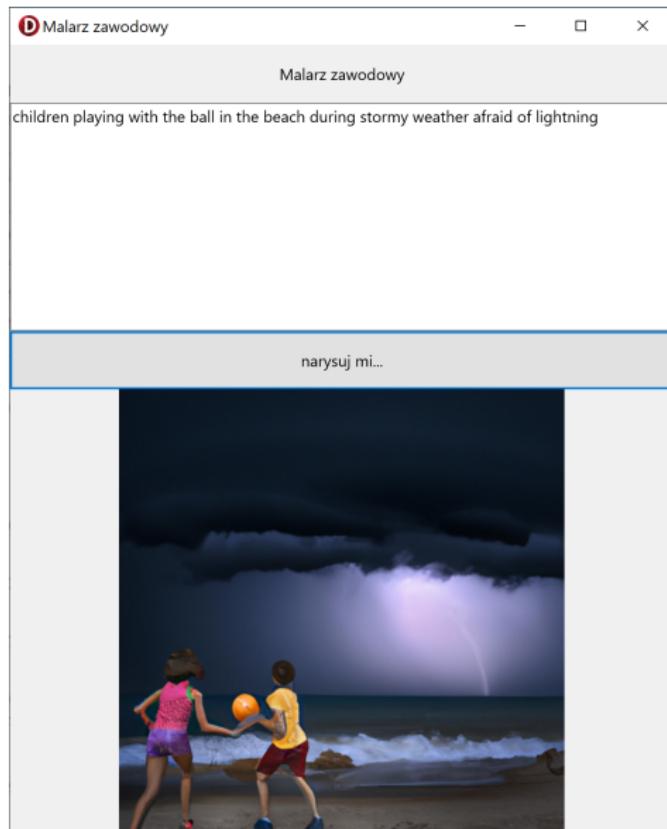


marcowe seminarium online

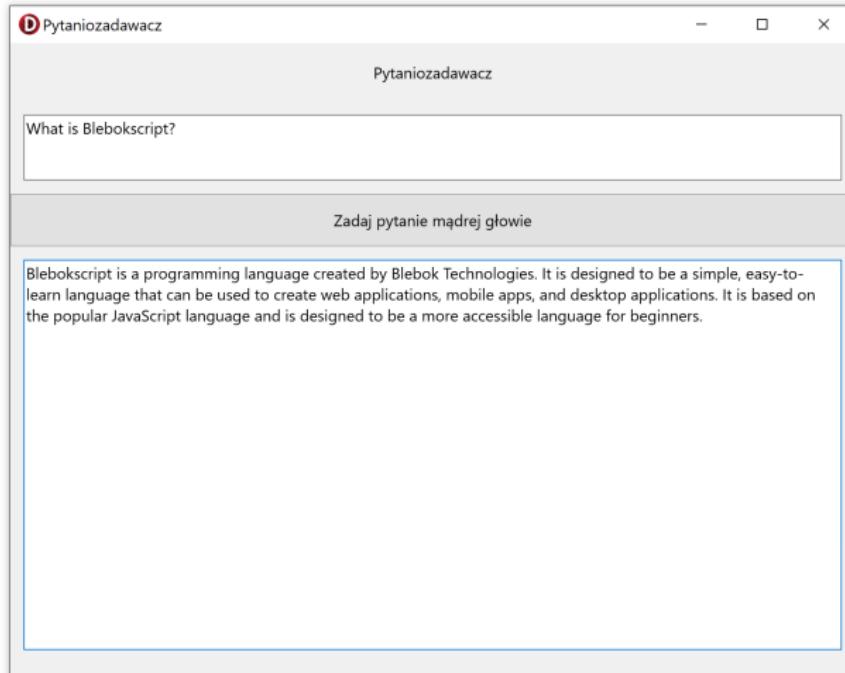
OpenAI (ChatGPT)

- ▶ różne modele (tekst, grafika, funkcje specjalne)
- ▶ REST API <https://api.openai.com/v1/>
- ▶ Przykład 1: text-davinci-003 zapytanie w języku naturalnym (synchronicznie)
- ▶ Przykład 2: tłumaczenie kodu na inny język programowania (asynchronicznie)
- ▶ Przykład 3: [https://api.openai.com/v1/images/generowanie grafiki](https://api.openai.com/v1/images/generowanie%20grafiki) (z kodu)
- ▶ Przykład 4: text-davinci-003 za pomocą biblioteki DelphiOpenAI (asynchronicznie TTask)

OpenAI – przykłady (1)



OpenAI – przykłady (2)



OpenAI – przykłady (3)

The screenshot shows a Windows application window titled "Pytaniozadawacz". The main area contains a question: "Give an example of script written in Blebokscript and point the main differences between Blebokscript and Javascript." Below this is a text input field labeled "Zadaj pytanie mądrzej głowie". The code editor area displays the following Blebokscript code:

```
// Blebokscript example

func main() {
    let x = 5;
    let y = 10;
    let z = x + y;
    print("The sum of x and y is: " + z);
}

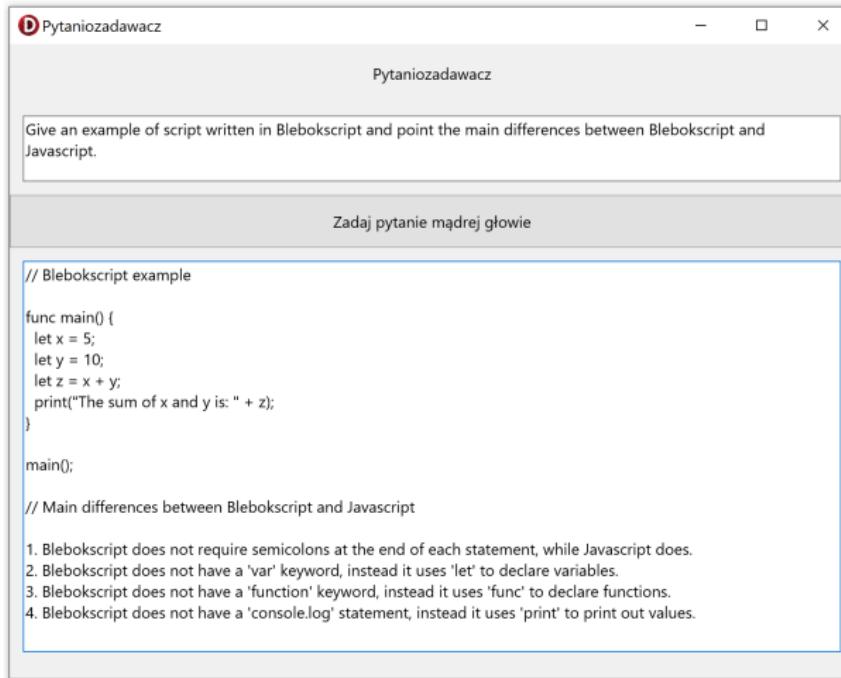
main();

// Main differences between Blebokscript and Javascript
```

Below the code, a list of differences is provided:

1. Blebokscript is a statically typed language, while Javascript is a dynamically typed language.
2. Blebokscript requires the use of the "func" keyword to declare functions, while Javascript does not.
3. Blebokscript does not have a "this" keyword, while Javascript does.
4. Blebokscript does not have a "for...in" loop, while Javascript does.
5. Blebokscript does not have a "switch" statement, while Javascript does.

OpenAI – przykłady (4)



OpenAI – przykłady (5)

The screenshot shows a web-based application window titled "Pytaniozadawacz". At the top, there is a header bar with the title and standard window controls (minimize, maximize, close). Below the header is a text input field containing the question: "Describe the development of modern mobile applications for iOS with the use of Algol.". A blue button labeled "Zadaj pytanie mądrzej głowie" (Ask a smarter question) is positioned below the input field. The main content area contains three paragraphs of text explaining the process of developing mobile apps for iOS using Algol. The text discusses the procedural nature of Algol, its use in writing step-by-step programs, and its power in creating various applications, including mobile ones. It then details the steps developers take: creating a user interface with the Cocoa Touch framework, writing code in Algol, and finally using Xcode to compile and release the application to the App Store.

Pytaniozadawacz

Pytaniozadawacz

Describe the development of modern mobile applications for iOS with the use of Algol.

Zadaj pytanie mądrzej głowie

Algol is a programming language that was developed in the 1950s and is still used today. It is a procedural language, meaning that it is designed to be used for writing programs that can be executed step-by-step. Algol is a powerful language that is used for a variety of applications, including mobile applications for iOS.

When developing mobile applications for iOS with Algol, developers must first create the application's user interface. This is done using the Cocoa Touch framework, which is a set of APIs that allow developers to create user interfaces for iOS applications. Once the user interface is created, developers can then use Algol to write the code that will power the application. Algol is a powerful language that allows developers to create complex applications with a variety of features.

Once the application is written, developers can then use the Xcode development environment to compile the code and create an executable application. Xcode is a powerful development environment that allows developers to debug their applications and test them on a variety of devices. Once the application is tested and ready for release, developers can then submit it to the App Store for approval.

In conclusion, Algol is a powerful language that can be used to develop modern mobile applications for iOS. Developers must first create the user interface using the Cocoa Touch framework, and then use Algol to write the code that will power the application. Once the application is written, developers can then use Xcode to compile the code and create an executable application. Finally, developers can submit the application to the App Store for approval.

OpenAI – przykłady (6)

The screenshot shows a Microsoft Edge browser window with the title "Pytaniozadawacz". The main content area contains a question: "Describe the development of modern mobile applications for Apple iOS with the use of BASIC interpreter and Commodore 128 computer and the deployment on iPhone devices." Below the question, a blue header bar says "Zadaj pytanie madrej głowie". The AI-generated answer is as follows:

The development of modern mobile applications for Apple iOS with the use of BASIC interpreter and Commodore 128 computer is a complex process. The first step is to create the application using the BASIC interpreter. This involves writing the code in the BASIC language and then compiling it into a program that can be run on the Commodore 128 computer. Once the program is written and compiled, it must be tested and debugged to ensure that it works correctly.

Once the program is ready, it must be deployed on the iPhone device. This requires the use of a special tool called Xcode, which is available for free from Apple. Xcode is used to create the necessary files and packages that are needed to deploy the application on the iPhone. Once the application is deployed, it can be tested on the device and any necessary changes can be made.

Finally, the application must be submitted to the Apple App Store for approval. Once approved, the application will be available for download by users.

Overall, the development of modern mobile applications for Apple iOS with the use of BASIC interpreter and Commodore 128 computer is a complex process that requires a great deal of knowledge and experience. However, with the right tools and resources, it is possible to create a successful application that can be deployed on the iPhone.

OpenAI – przykłady (7)

The screenshot shows a web-based AI interface. At the top, there's a header with the text "Pytaniozadawacz" and a logo. Below the header is a text input field containing the question: "Describe the development of modern mobile applications for Apple iOS with the use of ZX81 microcomputer and elaborate the benefits of 16kB RAM expansion." A blue button labeled "Zadaj pytanie mającej głowie" (Ask the question of the wise head) is positioned below the input field. The main content area contains three paragraphs of text. The first paragraph discusses the ZX81 as a home computer released in 1981, noting it was the first computer sold in the UK for less than £100, with 1K of RAM and a Z80 processor running at 3.25MHz. The second paragraph states that while the ZX81 was not designed for mobile app development, its low cost makes it a suitable platform. The third paragraph highlights the advantage of expandability, mentioning that up to 16K of RAM can be added, allowing for more complex applications like games.

Pytaniozadawacz

Describe the development of modern mobile applications for Apple iOS with the use of ZX81 microcomputer and elaborate the benefits of 16kB RAM expansion.

Zadaj pytanie mającej głowie

The development of modern mobile applications for Apple iOS with the use of ZX81 microcomputer is a relatively new concept. The ZX81 was a home computer released in 1981 by Sinclair Research Ltd. It was the first computer to be sold in the UK for less than £100. It was a very basic machine, with only 1K of RAM and a Z80 processor running at 3.25MHz.

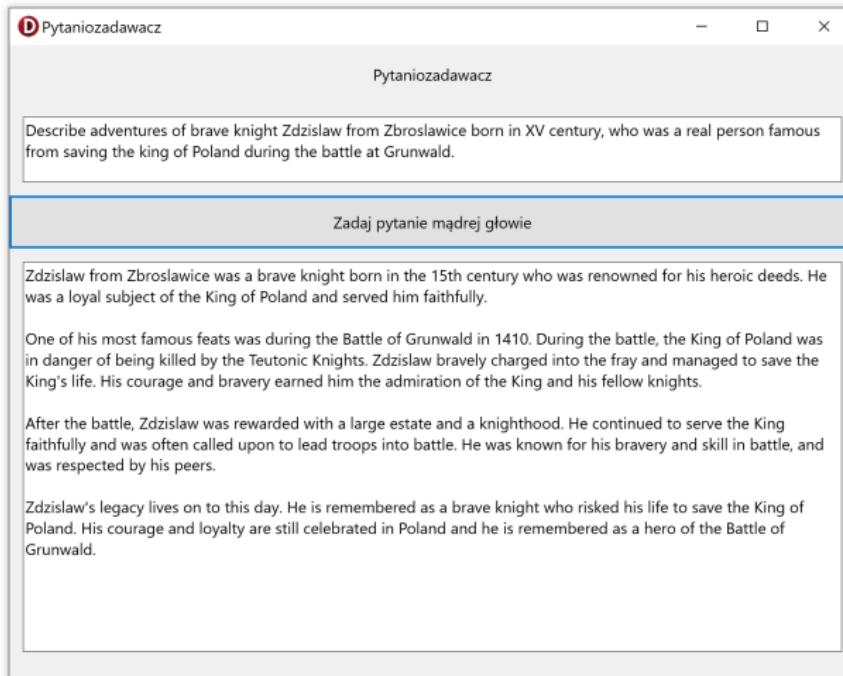
The ZX81 was not designed for mobile application development, but it can be used to create basic applications for Apple iOS. The main benefit of using the ZX81 for mobile application development is its low cost. The ZX81 is much cheaper than other development platforms, such as the Apple Mac or Windows PC.

The ZX81 also has the advantage of being able to be expanded with additional RAM. The ZX81 can be expanded up to 16K of RAM, which is enough to create basic applications for Apple iOS. This additional RAM allows developers to create more complex applications, such as games and other interactive applications.

The ZX81 also has the advantage of being able to be programmed in BASIC, which is a relatively simple programming language. This makes it easier for developers to create applications for Apple iOS.

Overall, the ZX81 is a great platform for developing basic applications for Apple iOS. It is inexpensive, expandable, and easy to program. The 16K RAM expansion is especially beneficial, as it allows developers to create more complex applications.

OpenAI – przykłady (8)



OpenAI – przykłady (9)

The screenshot shows a Microsoft Edge browser window with the title bar 'Pytaniozadawacz'. The main content area contains a question in a blue-bordered box: 'Compare in at least three paragraphs main characters from Hamlet with the outstanding Polish romance Life of Peter the Sausage.' Below this is a grey bar with the text 'Zadaj pytanie mądrzej głowie'. The main response is presented in a large white box with a black border. It starts by noting that 'Hamlet and Life of Peter the Sausage are two very different works of literature, but they both feature protagonists who are struggling to find their place in the world.' It then compares the main characters: 'The main character of Hamlet is the titular prince of Denmark. He is a complex and conflicted character who is struggling to come to terms with the death of his father and the subsequent marriage of his mother to his uncle. He is a thoughtful and introspective character who is often torn between his desire for revenge and his reluctance to act. He is a tragic figure who ultimately meets his demise at the hands of his own uncle.' followed by 'The main character of Life of Peter the Sausage is Peter, a young Polish sausage maker. He is a simple and humble character who is content with his life and his work. He is a kind and generous man who is always willing to help those in need. He is a loyal friend and a devoted husband and father. He is a symbol of hope and optimism in a world filled with despair and darkness. He ultimately finds his place in the world and is able to make a difference in the lives of those around him.' Finally, it concludes with 'Both Hamlet and Peter the Sausage are characters who are struggling to find their place in the world. Hamlet is a complex and conflicted character who is ultimately unable to find his place in the world and meets his demise. Peter the Sausage is a simple and humble character who is able to find his place in the world and make a difference in the lives of those around him. Both characters are symbols of hope and optimism in a world filled with despair and darkness.'

OpenAI – przykłady (10)

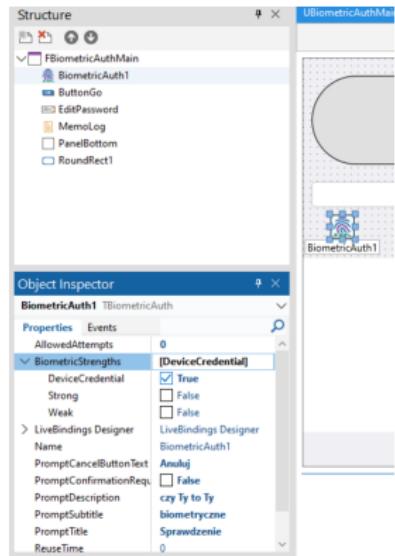
The screenshot shows a Microsoft Edge browser window with the title bar 'Pytaniozadawacz'. The main content area contains a text box with the following prompt:
Compare in at least three paragraphs main characters from Hamlet with the outstanding Polish romance Life of Peter the Sausage. Proof that Life of Peter the Sausage is an actual XIX-century decent literature.

Below the text box is a blue button labeled 'Zadaj pytanie mądrzej głowie' (Ask the smarter question). The page content is divided into several sections:

- Hamlet and Peter the Sausage**: Both characters are from different works of literature. Hamlet is a tragedy by William Shakespeare, while Peter the Sausage is a Polish romance by Józef Ignacy Kraszewski. Despite their differences, they share some similarities.
- Respective Stories**: Both characters are on a quest for justice and truth. Hamlet is trying to avenge his father's death, while Peter the Sausage is trying to uncover the truth about his family's past. Both are driven by a strong sense of morality and justice.
- Personalities**: Hamlet is complex and introspective, while Peter the Sausage is straightforward and optimistic. Hamlet is often seen as a tragic figure, while Peter the Sausage is seen as a more lighthearted and humorous character.
- Life of Peter the Sausage**: It is an actual XIX-century decent literature, first published in 1867. It has been translated into multiple languages and is considered one of the most important works of Polish literature from the 19th century. It explores themes like justice, morality, and family, and has been praised for its symbolism and exploration of the human condition.

TBiometricAuth

- ▶ dla iOS i Android
- ▶ konieczne włączenie uprawnień
- ▶ dla Android konieczne ustawienie property BiometricStrengths
- ▶ zdarzenia:
 - OnAuthenticationFail
 - OnAuthenticationSuccess



TBiometricAuth – ustawienia Android

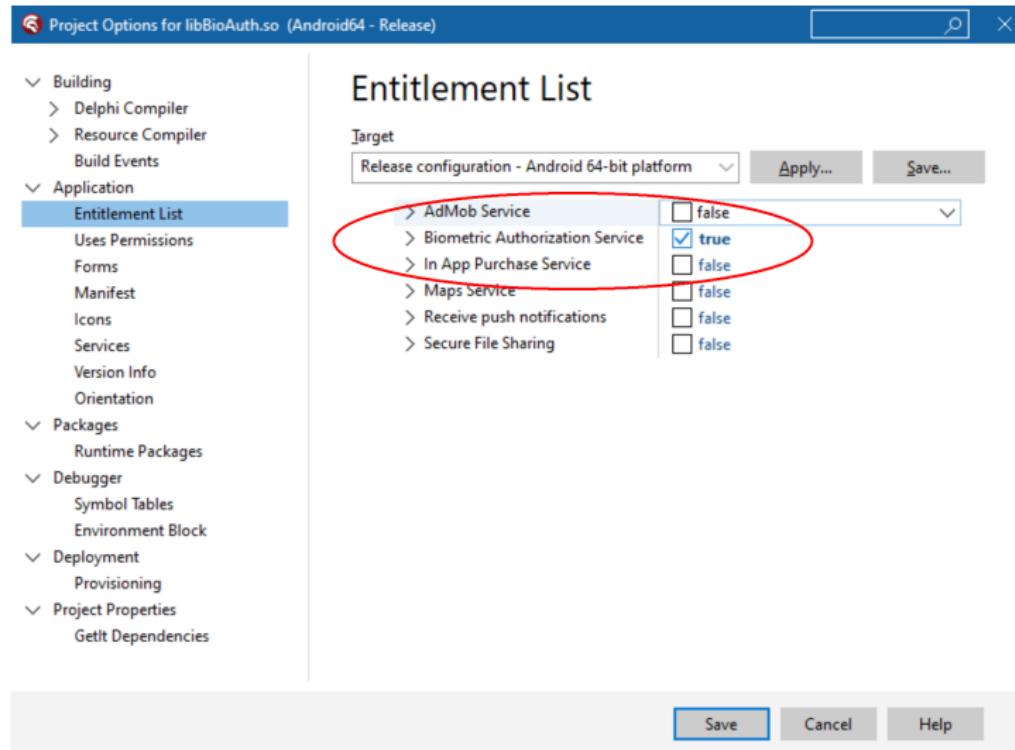
Project Options for libBioAuth.so (Android64 - Release)

Entitlement List

Target: Release configuration - Android 64-bit platform

Entitlement	Value
> AdMob Service	<input type="checkbox"/> false
> Biometric Authorization Service	<input checked="" type="checkbox"/> true
> In App Purchase Service	<input type="checkbox"/> false
> Maps Service	<input type="checkbox"/> false
> Receive push notifications	<input type="checkbox"/> false
> Secure File Sharing	<input type="checkbox"/> false

Save Cancel Help



TBiometricAuth – uprawnienia Android

Project Options for libBioAuth.so (Android64 - Release)

Building

- > Delphi Compiler
- > Resource Compiler

Build Events

Application

- Entitlement List
- Uses Permissions**
- Forms
- Manifest
- Icons
- Services
- Version Info
- Orientation

Packages

- Runtime Packages

Debugger

- Symbol Tables
- Environment Block

Deployment

- Provisioning

Project Properties

- GetIt Dependencies

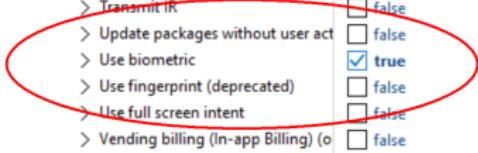
Uses Permissions

Target: Release configuration - Android 64-bit platform

Apply... Save...

> Schedule exact alarm	<input type="checkbox"/> false
> Set alarm	<input type="checkbox"/> false
> Set wallpaper	<input type="checkbox"/> false
> Set wallpaper hints	<input type="checkbox"/> false
> Subscribed feeds read (obsolete)	<input type="checkbox"/> false
> Transmit IR	<input type="checkbox"/> false
> Update packages without user act	<input type="checkbox"/> false
> Use biometric	<input checked="" type="checkbox"/> true
> Use fingerprint (deprecated)	<input type="checkbox"/> false
> Use full screen intent	<input type="checkbox"/> false
> Vending billing (In-app Billing) (o	<input type="checkbox"/> false
> Vibrate	<input type="checkbox"/> false
> Wake lock	<input type="checkbox"/> false
> Write sync settings	<input type="checkbox"/> false
> Write user dictionary (obsolete)	<input type="checkbox"/> false
Dangerous (runtime user approval)	
> Access background location	<input type="checkbox"/> false
> Access coarse location	<input checked="" type="checkbox"/> true

Save Cancel Help



TBiometricAuth – uprawnienia iOS

The screenshot shows the 'Version Info' section of the Delphi Project Options dialog for an iOS project. The left sidebar lists various project settings like Building, Application, and Version Info (which is selected). The main area shows the 'Target' set to 'Release configuration - iOS Device 64-bit platform'. A checked checkbox 'Include version information in project' is present. Below it, 'Module version number' fields for Major (1), Minor (0), and Build (0) are shown. Under 'Build number options', a dropdown menu is set to 'Do not change build number'. A red oval highlights a table of key-value pairs under 'Key' and 'Value'. The table includes NSPhotoLibraryUsageDescription, NSPhotoLibraryAddUsageDescription, NSCameraUsageDescription, NSFacelIDUsageDescription, NSMicrophoneUsageDescription, NS SiriUsageDescription, ITSAppUsesNonExemptEncryption, NSBluetoothAlwaysUsageDescription, and SBluetoothPeripheralUsageDescription.

Key	Value
NSPhotoLibraryUsageDescription	The reason for accessing the photo libra
NSPhotoLibraryAddUsageDescription	The reason for adding to the photo libra
NSCameraUsageDescription	The reason for accessing the camera
NSFacelIDUsageDescription	Bedziem patrzec w oczy :)
NSMicrophoneUsageDescription	The reason for accessing the microphone
NS SiriUsageDescription	The reason for accessing Siri
ITSAppUsesNonExemptEncryption	false
NSBluetoothAlwaysUsageDescription	The reason for accessing bluetooth
SBluetoothPeripheralUsageDescription	The reason for accessing bluetooth perif

Szlif dla iOS

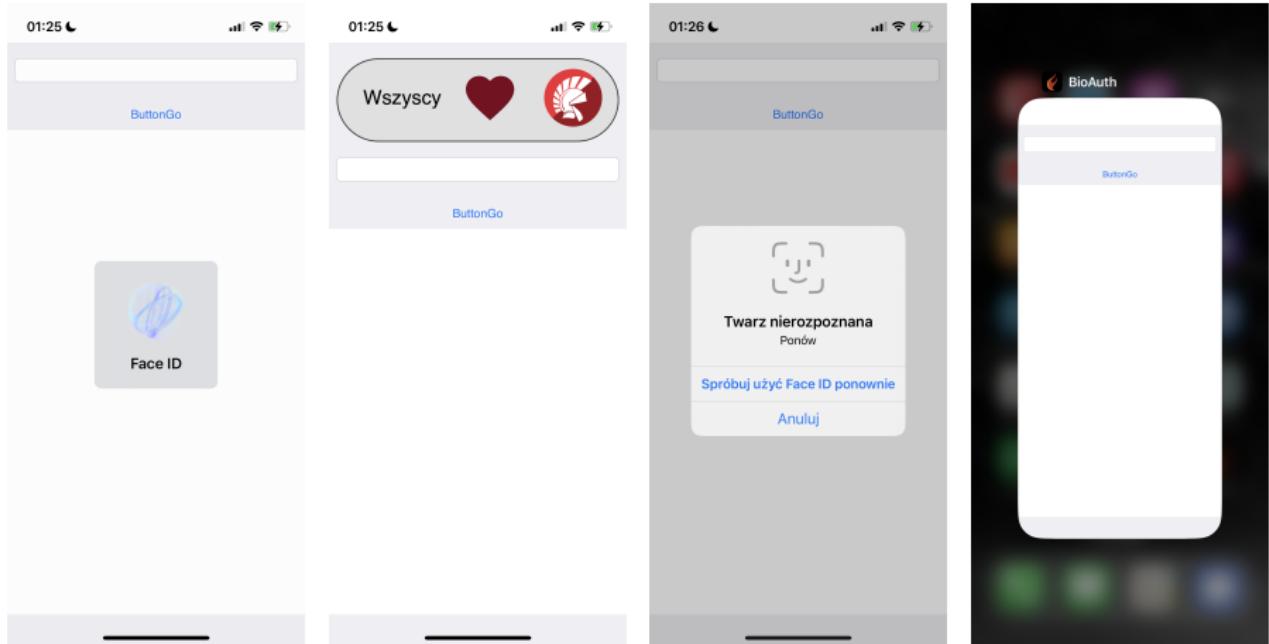
- ▶ ukrywanie sekretu na podglądzie

```
WindowHandleToPlatform(Handle).View.←  
    setNeedsDisplay;
```

- ▶ rozróżnianie modeli z notch

```
var KeyWindow := SharedApplication.keyWindow;  
if (KeyWindow <> nil) and (KeyWindow.rootViewController <> nil) ←  
    then  
begin  
    PanelBottom.Visible := TOSVersion.Check(11) and (KeyWindow.←  
        safeAreaInsets.bottom > 0);  
end;
```

TBiometricAuth/FaceID – iOS/iPhone 13



(a) FacID w
działaniu

(b) Sekret odkryty

(c) Niepowodzenie

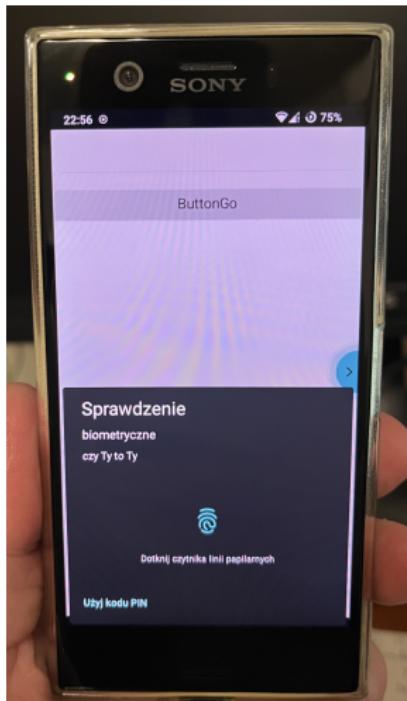
(d) Sekret ukryty
w podglądzie

Szlif dla Android

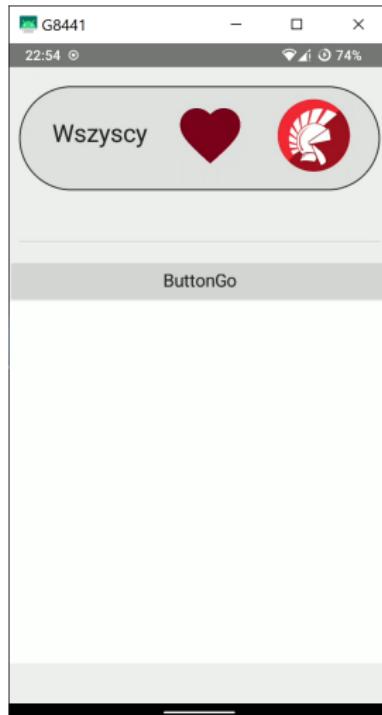
- ▶ ukrywanie sekretu na podglądzie
(FMX.Platform.Android.pas)

```
procedure TPlatformAndroid.HandleApplicationCommandEvent(const AAppGlue: TAndroidApplicationGlue; const ACommand: TAndroidApplicationCommand);  
begin  
  case ACommand of  
    TAndroidApplicationCommand.LostFocus: //zs2023  
      begin  
        MainActivity.getWindow.setFlags( 8192 , 8192 );  
        (Application.MainForm as TFBiometricAuthMain).HideSecret;  
      end;  
    TAndroidApplicationCommand.GainedFocus: //zs2023  
      begin  
        MainActivity.getWindow.clearFlags( 8192 );  
        (Application.MainForm as TFBiometricAuthMain).←  
          RequestBiometricAuth;  
      end;  
  end;  
  (...)
```

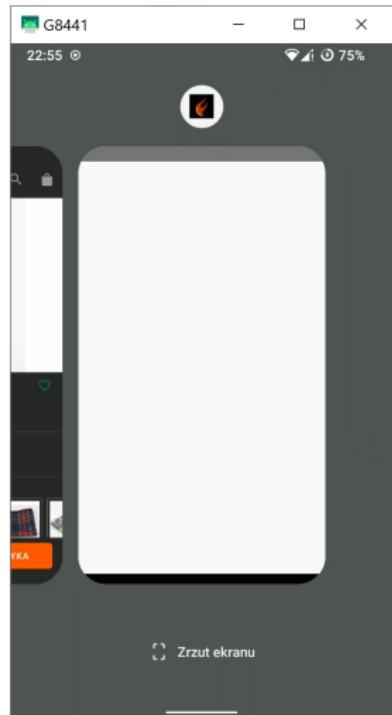
TBiometricAuth – Android



(a) Android 12 – dialog systemowy



(b) Sekret odkryty



(c) Sekret ukryty w podglądzie

Dziękuję za uwagę

pytania: zdzislaw@sroczynski.pl