2007

2019






# Programowanie dla iOS




# Katalog Lego

# Gdzie kupić klocki 1?

# Gdzie kupić klocki I?

Zabawki i gry > Klocki > Klocki plastikowe > Miasto




Powiększ

## Lego Creator, Dom na wzgórzu, 5771, klocki

( 0 ocen) Sprawdź recenzje

| | |
|---|---|
| Marka: | Lego |
| Płeć: | chłopiec / dziewczynka |
| Przedział wiekowy: | 8-11 lat |
| Ilość elementów: | 714 |

darmowa dostawa

Szczegółowe informacje dotyczące cennika i warunków dostawy znajdziesz tutaj. Zapraszamy!

cena: 269,99 zł
~~309,00 zł~~
do koszyka
dodaj do schowka
Wyślemy w 24h

poleć znajomemu
Udostepnij na:

# Gdzie kupić klocki I?

Zabawki i gry > Klocki > Klocki plastikowe > Miasto




Powiększ

## Lego Creator, Dom na wzgórzu, 5771, klocki

( 0 ocen) Sprawdź recenzje

| | |
|---|---|
| Marka: | Lego |
| Płeć: | chłopiec / dziewczynka |
| Przedział wiekowy: | 8-11 lat |
| Ilość elementów: | 714 |

darmowa dostawa

Szczegółowe informacje dotyczące cennika i warunków dostawy znajdziesz tutaj. Zapraszamy!

cena: 269,99 zł
~~309,00 zł~~
do koszyka
dodaj do schowka
Wyślemy w 24h

poleć znajomemu
Udostepnij na:

# Gdzie kupić klocki I?

Zabawki i gry > Klocki > Klocki plastikowe > Miasto




Powiększ

## Lego Creator, Dom na wzgórzu, 5771, klocki

( 0 ocen) Sprawdź recenzje

Marka: Lego
Płeć: chłopiec / dziewczynka
Przedział wiekowy: 8-11 lat
Ilość elementów: 714

darmowa dostawa

Szczegółowe informacje dotyczące cennika i warunków dostawy znajdziesz tutaj. Zapraszamy!

cena: 269,99 zł
~~309,00 zł~~
do koszyka
dodaj do schowka
Wyślemy w 24h

poleć znajomemu
Udostepnij na:

LEGO CREATOR

## Hillside House

Item: **5771**
Ages:**8-12**
Pieces:**714**
VIP Points: **65**
Available Now

Build your dream home with 3-in-1 style!

Price **zł 259.99**

# Gdzie kupić klocki I?

Zabawki i gry > Klocki > Klocki plastikowe > Miasto




Powiększ

## Lego Creator, Dom na wzgórzu, 5771, klocki

( 0 ocen) Sprawdź recenzje

Marka: Lego
Płeć: chłopiec / dziewczynka
Przedział wiekowy: 8-11 lat
Ilość elementów: 714

darmowa dostawa

Szczegółowe informacje dotyczące cennika i warunków dostawy znajdziesz tutaj. Zapraszamy!

cena: 269,99 zł
~~309,00 zł~~
do koszyka
dodaj do schowka
Wyślemy w 24h

poleć znajomemu
Udostepnij na:

LEGO CREATOR

## Hillside House

Item: 5771
Ages:8-12
Pieces:714

VIP Points: 65
Available Now

Build your dream home with 3-in-1 style!

Price zł 259.99

## LEGO 5771 Creator dom na wzgórzu

Cena produktu: **215,00 zł**

Do schowka

Zamawiana ilość: 1 Dodaj do koszyka




Kliknij aby powiększyć

| | |
|---|---|
| **Numer katalogowy** | L.5771 |
| **Producent** | LEGO |
| **Stan magazynowy** | |

# Gdzie kupić klocki 2?

# Gdzie kupić klocki 2?

Lego 10220 Volkswagen T1 Camper



**Kup Teraz!** **515,00 zł**

Do końca: 7 dni Popularność: -

LEGO 10220 Volkswagen T1 Camper Van Nowość od ręki

manami

**Kup Teraz!** **599,95 zł**

Do końca: 7 dni Popularność: -

LEGO CREATOR 10220 Volkswagen T1 Camper Van WROCŁA




**Kup Teraz!** **545,00 zł**

Do końca: 8 dni Popularność: -

LEGO 10220 Volkswagen T1 C, UNIKAT,Wwa CH PANORAMA



**Kup Teraz!** **600,00 zł**

Do końca: 9 dni Popularność: -

LEGO CREATOR 10220 Volkswagen T1 Camper Van - FVAT




**Kup Teraz!** **545,00 zł**

Do końca: 9 dni Popularność: -

KLOCKI LEGO 10220 VOLKSWAGEN CAMPER OGÓREK RARYTAS



**Kup Teraz!** **789,99 zł**

Do końca: 29 dni Popularność: -

# Gdzie kupić klocki 2?

Lego 10220 Volkswagen T1 Camper



**Kup Teraz! 515,00 zł**

Do końca: 7 dni Popularność: -

LEGO 10220 Volkswagen T1 Camper Van Nowość od ręki

manami

**Kup Teraz! 599,95 zł**

Do końca: 7 dni Popularność: -

LEGO CREATOR 10220 Volkswagen T1 Camper Van WROCŁA




**Kup Teraz! 545,00 zł**

Do końca: 8 dni Popularność: -

LEGO 10220 Volkswagen T1 C, UNIKAT,Wwa CH PANORAMA



**Kup Teraz! 600,00 zł**

Do końca: 9 dni Popularność: -

LEGO CREATOR 10220 Volkswagen T1 Camper Van - FVAT




**Kup Teraz! 545,00 zł**

Do końca: 9 dni Popularność: -

KLOCKI LEGO 10220 VOLKSWAGEN CAMPER OGÓREK RARYTAS



**Kup Teraz! 789,99 zł**

Do końca: 29 dni Popularność: -

# Gdzie kupić klocki 2?

## Volkswagen T1 Camper Van

Item: 10220

Limit: 5 ?

Ages:16+

VIP Points: 110

Pieces:1332

Available Now

Blast back to 1962 and cruise into psychedelic style!

Cruise in psychedelic style with this classic 1962 Volkswagen Camper Van, featuring iconic 'splittie' safari windshield and pop-up roof!

Price zł 439.99

# Gdzie kupić klocki 3?

# Gdzie kupić klocki 3?

LEGO Exclusive 10231 Prom Kosmiczny /Świdnica /24h



**Kup Teraz!** **499,00 zł**

Do końca: 8 dni Popularność: -

LEGO EXCLUSIVE 10231 SHUTTLE EXPEDITION



**Kup Teraz!** **379,99 zł**

Do końca: 12 dni Popularność: -

KLOCKI LEGO EXCLUSIVE 10231 PROM KOSMICZNY Kraków




**Kup Teraz!** **549,99 zł**

Do końca: 13 dni Popularność: -

# Gdzie kupić klocki 3?

**LEGO 10231 Shuttle adventure (prom kosmiczny) 2011**

Cena produktu: **499,00 zł**

Do schowka

Zamawiana ilość: 1

Dodaj do koszyka



Kliknij aby powiększyć

| | |
|---|---|
| **Numer katalogowy** | L.10231 |
| **Producent** | LEGO |
| **Stan magazynowy** | |

# Gdzie kupić klocki 3?

## Shuttle Expedition

Item: 10231

Ages:14+ VIP Points: 95

Pieces:1230 Available Now

Blast off on an outer space mission!

Price zł 379.99

# Gdzie kupić klocki 4?

# Gdzie kupić klocki 4?



# Gdzie kupić klocki 4?



# Gdzie kupić klocki 4?



**LEGO 6131 Creator pudło klocków** LEGO

Cena: **128,01 zł**

1

LEGO CREATOR PUDŁO KLOCKÓW

# Gdzie kupić klocki 4?

| | Nazwa | Cena | z dostawą |
|---|---|---|---|
|  | LEGO 6131 CREATOR WIADERKO ZESTAW XXL | Kup Teraz! 98,00 zł | 110,00 zł |
|  | LEGO 6131 ZESTAW STARTOWY 600 KLOCKÓW + SKRZYNIA | Kup Teraz! 99,00 zł | 113,99 zł |
|  | LEGO CREATOR MAXI ZESTAW 6131 + SKRZYNIA | Kup Teraz! 99,99 zł | 116,99 zł |
|  | LEGO CREATOR MAXI ZESTAW 6131 + SKRZYNIA - KURIER | Kup Teraz! 99,99 zł | 116,99 zł |
|  | MEGA ZESTAW KLOCKÓW LEGO CREATOR 6131 kurier DPD | Kup Teraz! 109,00 zł | 129,00 zł |

# Katalog Lego

# Katalog Lego

- Informacje o zestawach przechowywane lokalnie w bazie danych

# Katalog Lego

- Informacje o zestawach przechowywane lokalnie w bazie danych
- Parser strony shop.lego.com pod Windows

# Katalog Lego

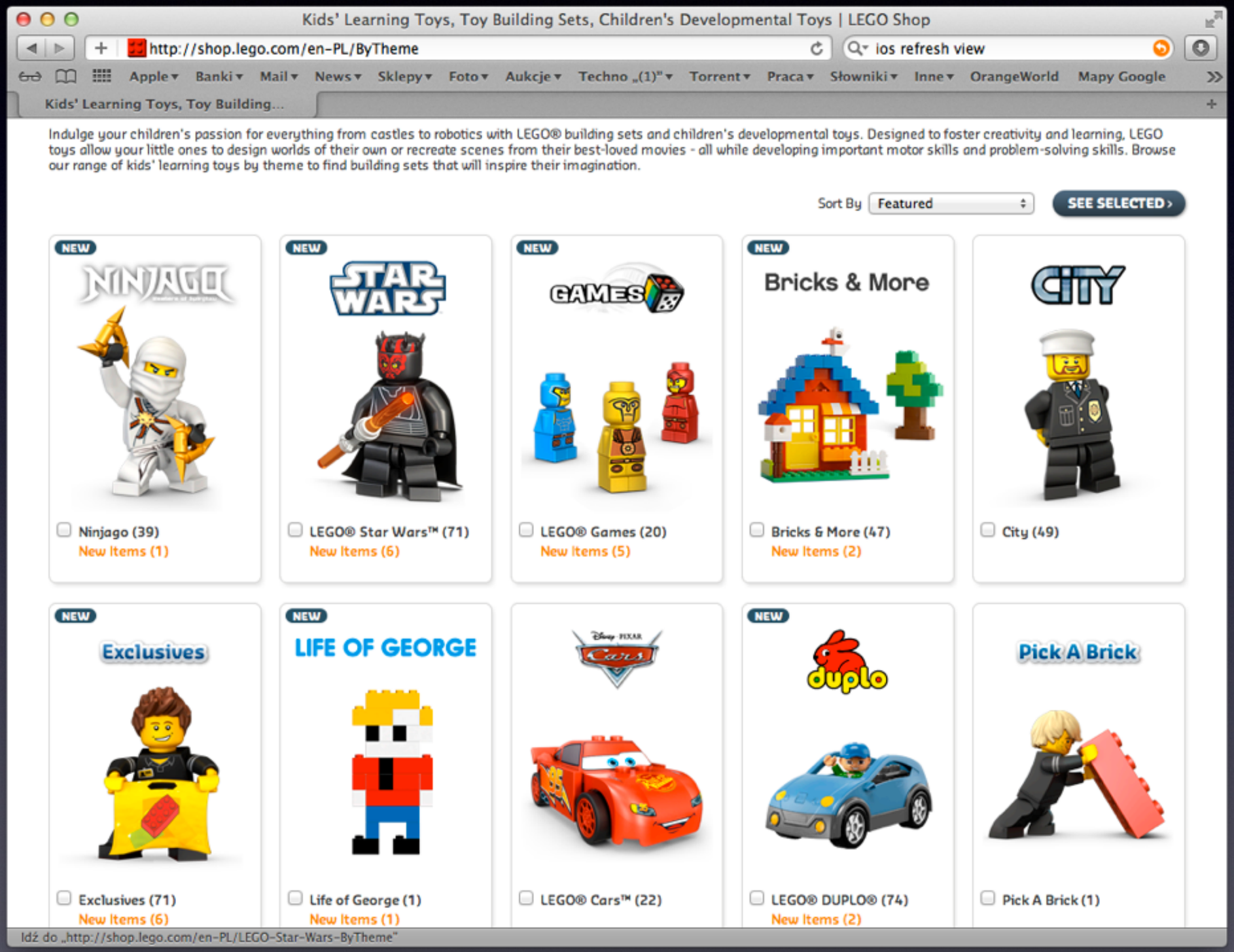
- Informacje o zestawach przechowywane lokalnie w bazie danych
- Parser strony shop.lego.com pod Windows
- Dane eksportowane do bazy sqlite

# Katalog Lego

- Informacje o zestawach przechowywane lokalnie w bazie danych
- Parser strony shop.lego.com pod Windows
- Dane eksportowane do bazy sqlite
- Ceny ze sklepu Lego i Allegro

# Strona Lego

# Strona Lego


Kids' Learning Toys, Toy Building Sets, Children's Developmental Toys | LEGO Shop
http://shop.lego.com/en-PL/ByTheme
ios refresh view
Apple Banki Mail News Sklepy Foto Aukcje Techno „(1)" Torrent Praca Słowniki Inne OrangeWorld Mapy Google
Kids' Learning Toys, Toy Building...
Indulge your children's passion for everything from castles to robotics with LEGO® building sets and children's developmental toys. Designed to foster creativity and learning, LEGO toys allow your little ones to design worlds of their own or recreate scenes from their best-loved movies - all while developing important motor skills and problem-solving skills. Browse our range of kids' learning toys by theme to find building sets that will inspire their imagination.
Sort By Featured
SEE SELECTED
NEW
NINJAGO
Ninjago (39)
New Items (1)
NEW
STAR WARS
LEGO® Star Wars™ (71)
New Items (6)
NEW
GAMES
LEGO® Games (20)
New Items (5)
NEW
Bricks & More
Bricks & More (47)
New Items (2)
CITY
City (49)
NEW
Exclusives
Exclusives (71)
New Items (6)
NEW
LIFE OF GEORGE
Life of George (1)
New Items (1)
Disney Pixar Cars
LEGO® Cars™ (22)
NEW
duplo
LEGO® DUPLO® (74)
New Items (2)
Pick A Brick
Pick A Brick (1)
Idź do „http://shop.lego.com/en-PL/LEGO-Star-Wars-ByTheme"

# Parser

# Parser

- Parsujemy zestawy wg kategorii

# Parser

- Parsujemy zestawy wg kategorii
- Jeżelli zestaw jest w kilku kategoriach zapamiętujemy to

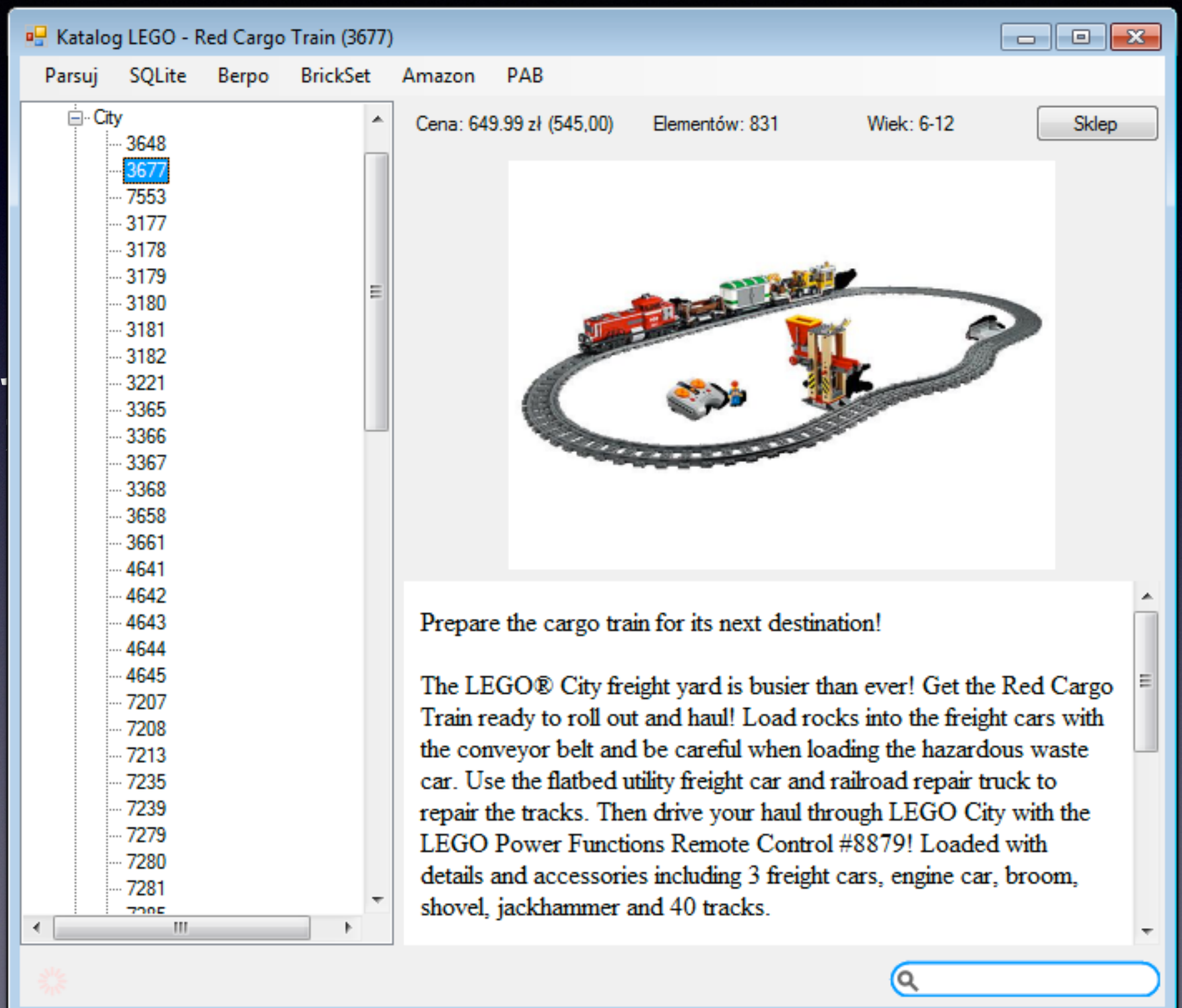
# Parser

- Parsujemy zestawy wg kategorii
- Jeżelli zestaw jest w kilku kategoriach zapamiętujemy to
- Opis zestawu jest pamiętany jako HTML

# Parser

- Parsujemy zestawy wg kategorii
- Jeżelli zestaw jest w kilku kategoriach zapamiętujemy to
- Opis zestawu jest pamiętany jako HTML
- Dodatkowo parsujemy polski sklep internetowy

# Parser


Katalog LEGO - Red Cargo Train (3677)
Parsuj
SQLite
Berpo
BrickSet
Amazon
PAB
City
3648
3677
7553
3177
3178
3179
3180
3181
3182
3221
3365
3366
3367
3368
3658
3661
4641
4642
4643
4644
4645
7207
7208
7213
7235
7239
7279
7280
7281
Cena: 649.99 zł (545,00)
Elementów: 831
Wiek: 6-12
Sklep
Prepare the cargo train for its next destination!
The LEGO® City freight yard is busier than ever! Get the Red Cargo Train ready to roll out and haul! Load rocks into the freight cars with the conveyor belt and be careful when loading the hazardous waste car. Use the flatbed utility freight car and railroad repair truck to repair the tracks. Then drive your haul through LEGO City with the LEGO Power Functions Remote Control #8879! Loaded with details and accessories including 3 freight cars, engine car, broom, shovel, jackhammer and 40 tracks.

SQLite



# SQLite




- Kompaktowa baza danych napisana w C

# SQLite




- Kompaktowa baza danych napisana w C
- Public Domain

# SQLite




- Kompaktowa baza danych napisana w C
- Public Domain
- implementuje większość funkcji z SQL-92

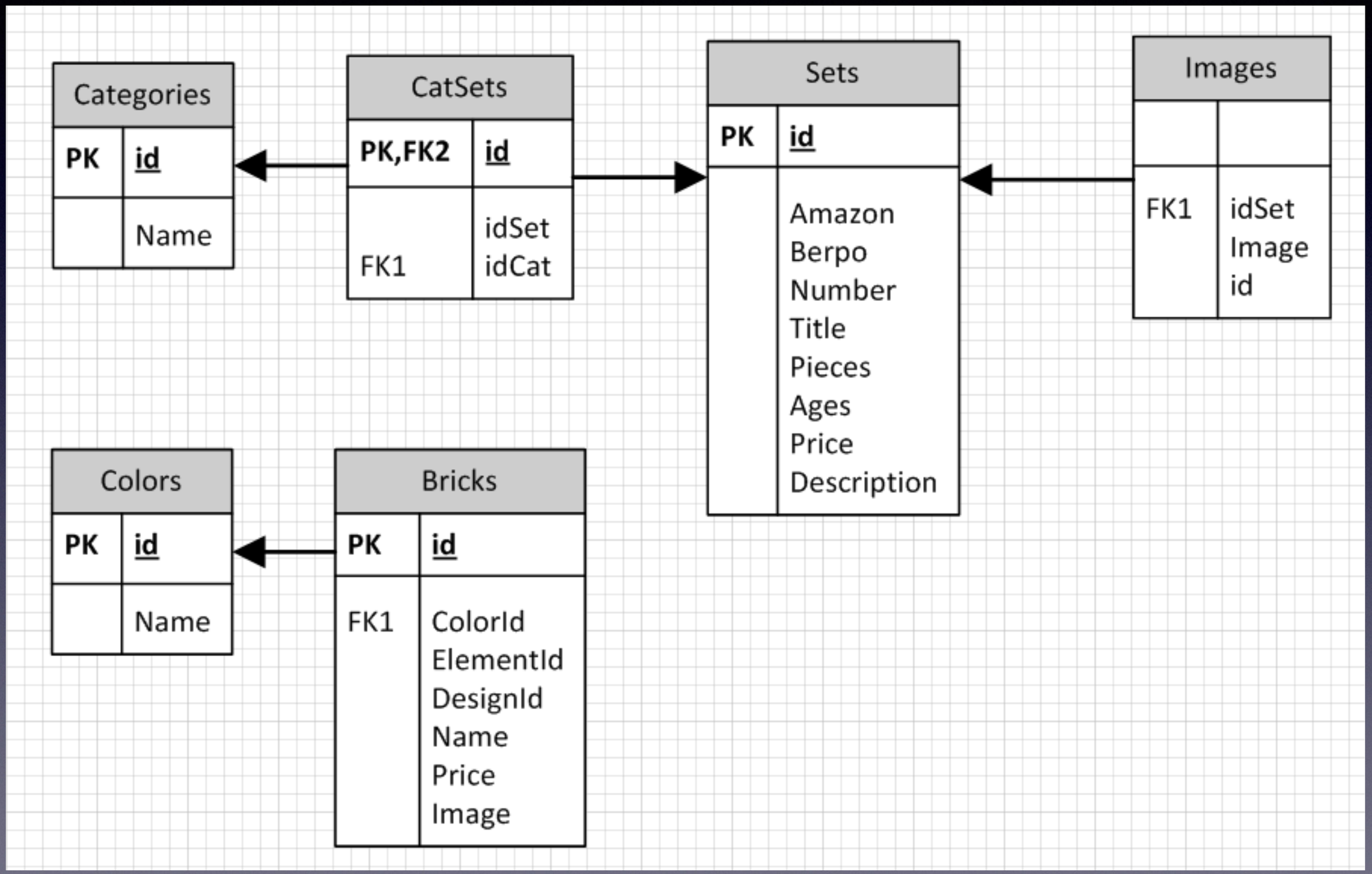
# SQLite




- Kompaktowa baza danych napisana w C
- Public Domain
- implementuje większość funkcji z SQL-92
- wykorzystywany w wielu systemach i aplikacjach

# Schemat bazy danych

# Schemat bazy danych



# Nowy projekt

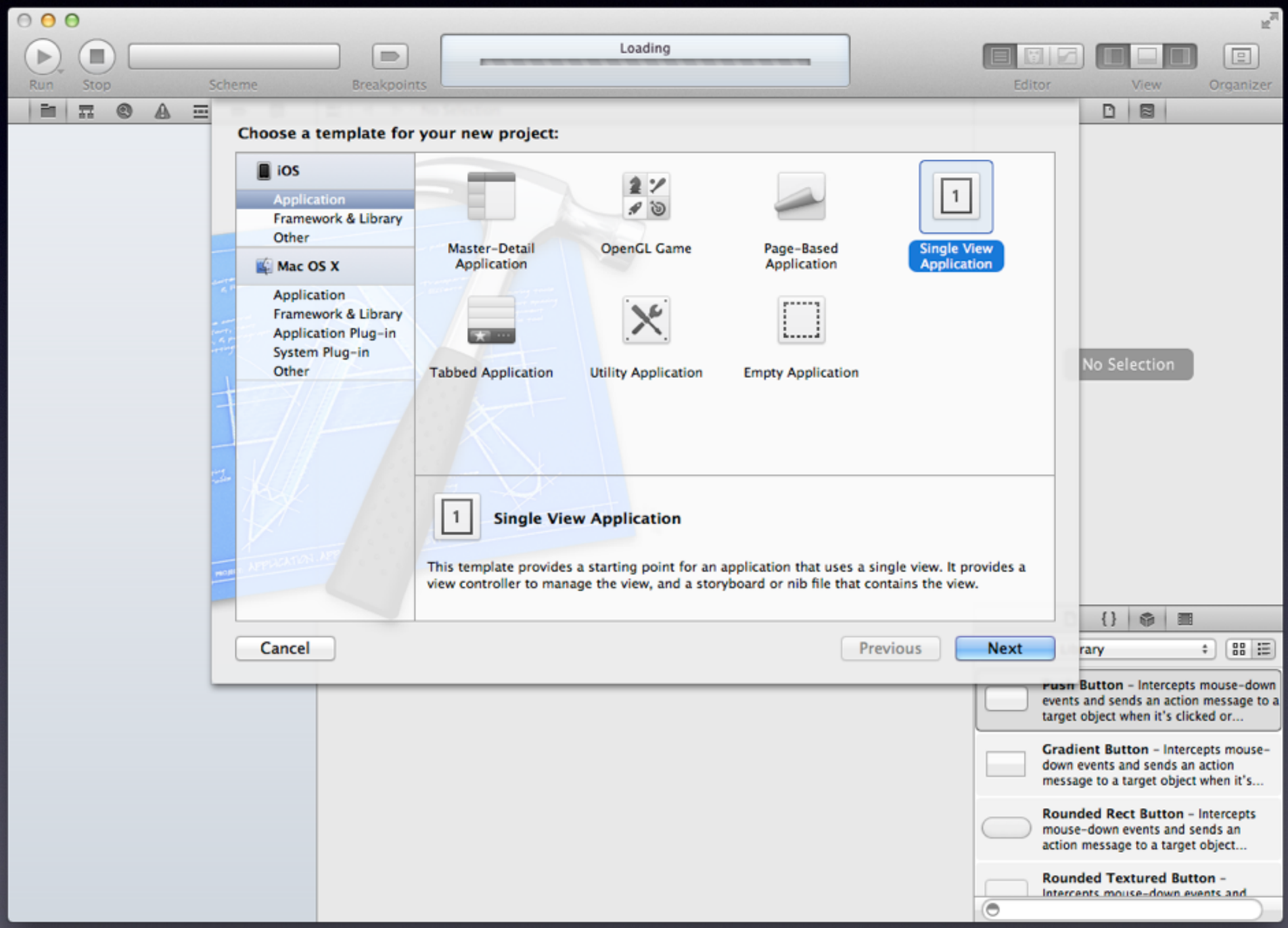
# Nowy projekt

- projekt w Xcode 4

# Nowy projekt

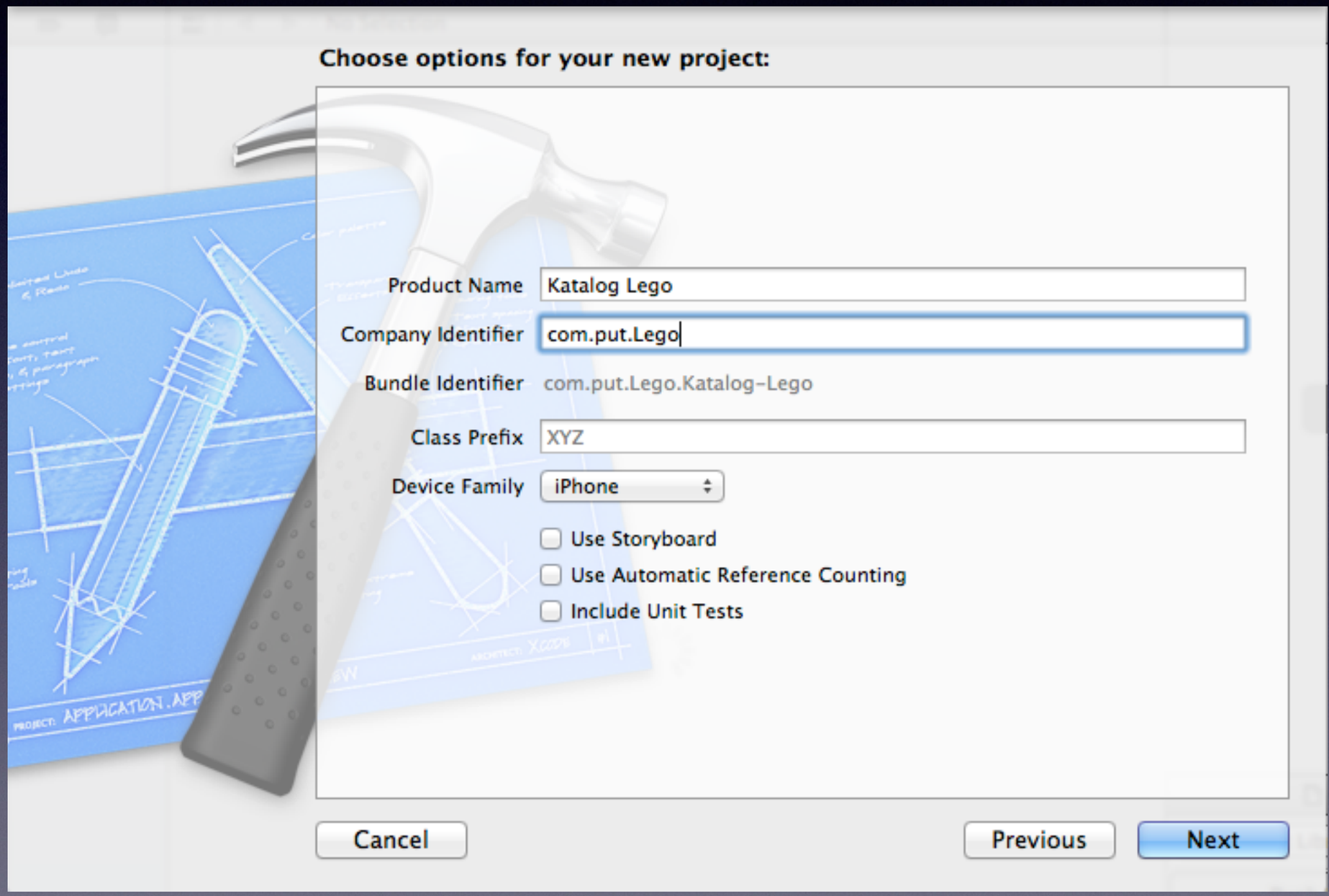
- projekt w Xcode 4
- aplikacja typu nawigacyjnego

# Nowy projekt



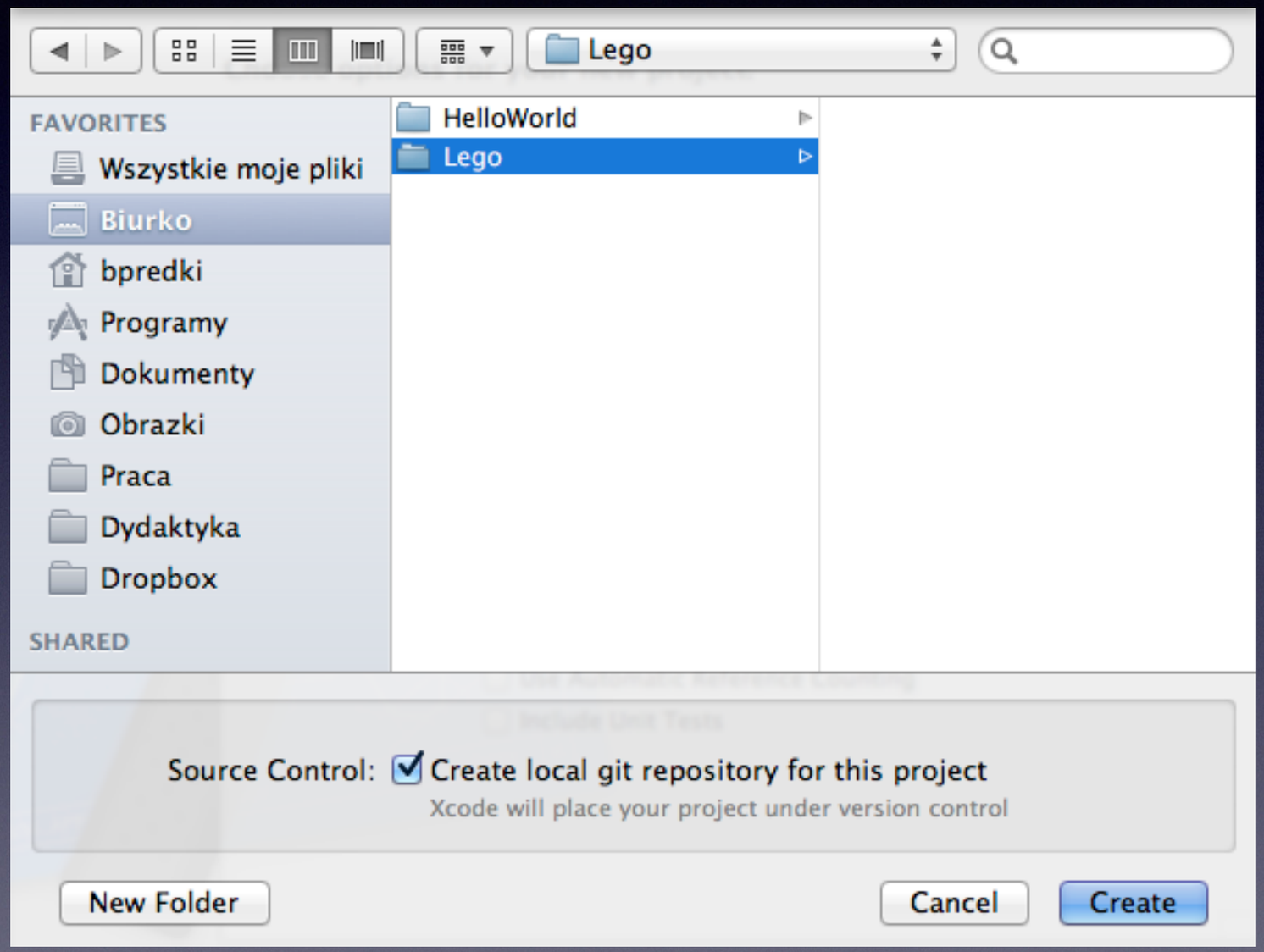
# Nowy projekt

# Nowy projekt



# Nowy projekt

# Nowy projekt



# Ustawienia projektu

# Ustawienia projektu

- Nazwa

# Ustawienia projektu

- Nazwa
- Docelowa platforma

# Ustawienia projektu

- Nazwa
- Docelowa platforma
- Ikony:

# Ustawienia projektu

- Nazwa
- Docelowa platforma
- Ikony:
  - zwykła 57x57 px

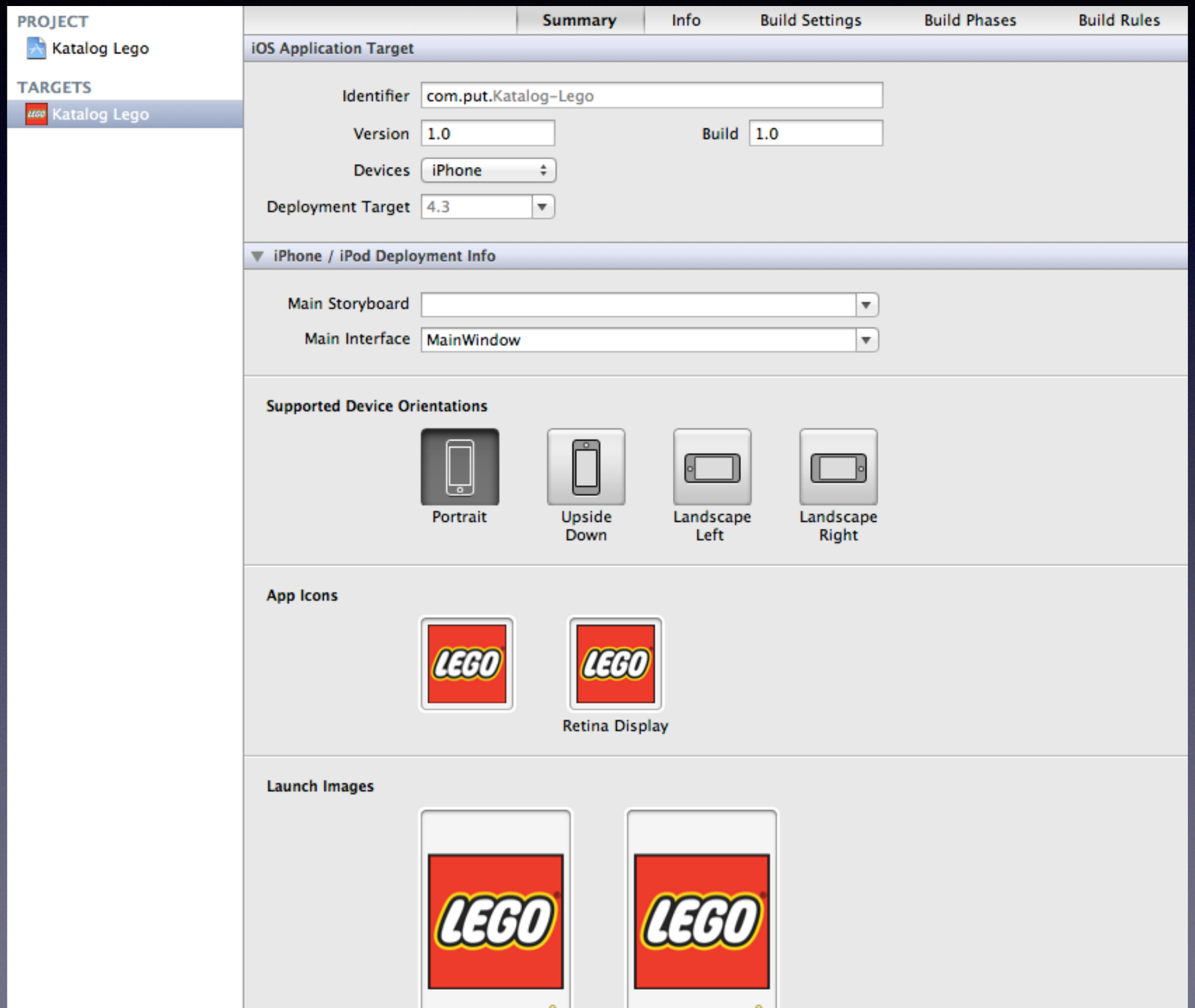
# Ustawienia projektu

- Nazwa
- Docelowa platforma
- Ikony:
  - zwykła 57x57 px
  - retina 114x114 px

# Ustawienia projektu

- Nazwa
- Docelowa platforma
- Ikony:
  - zwykła 57x57 px
  - retina 114x114 px
  - ekran startowy

# Ustawienia projektu



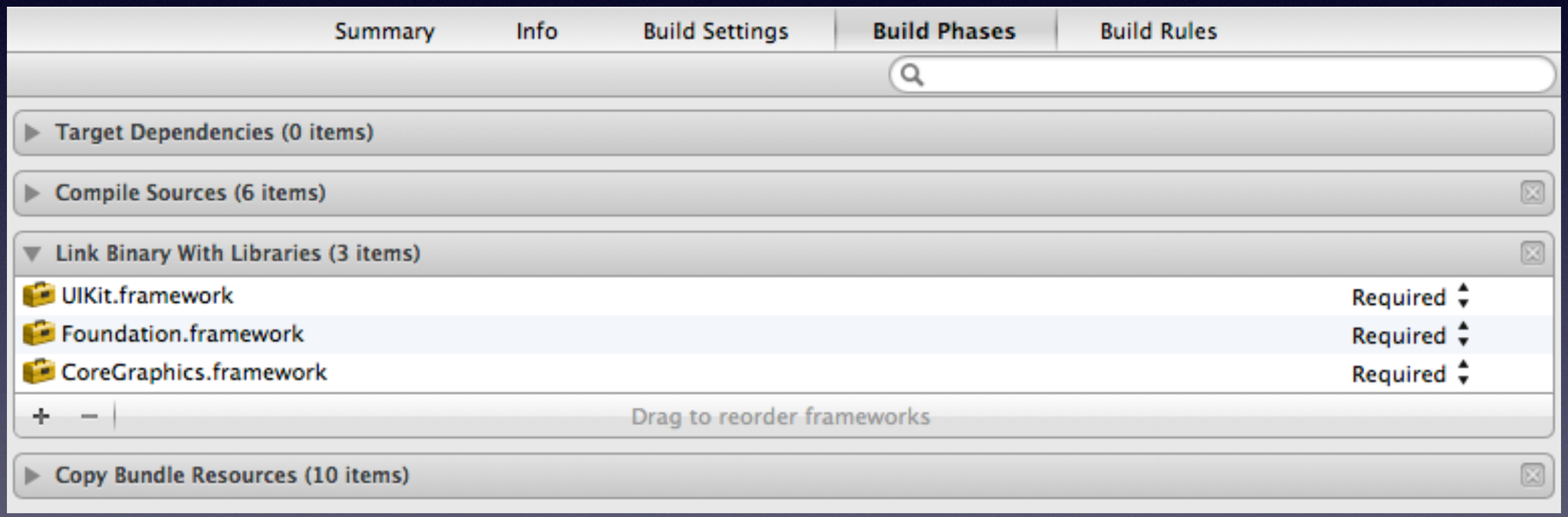
# Biblioteka SQLite

# Biblioteka SQLite

- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib

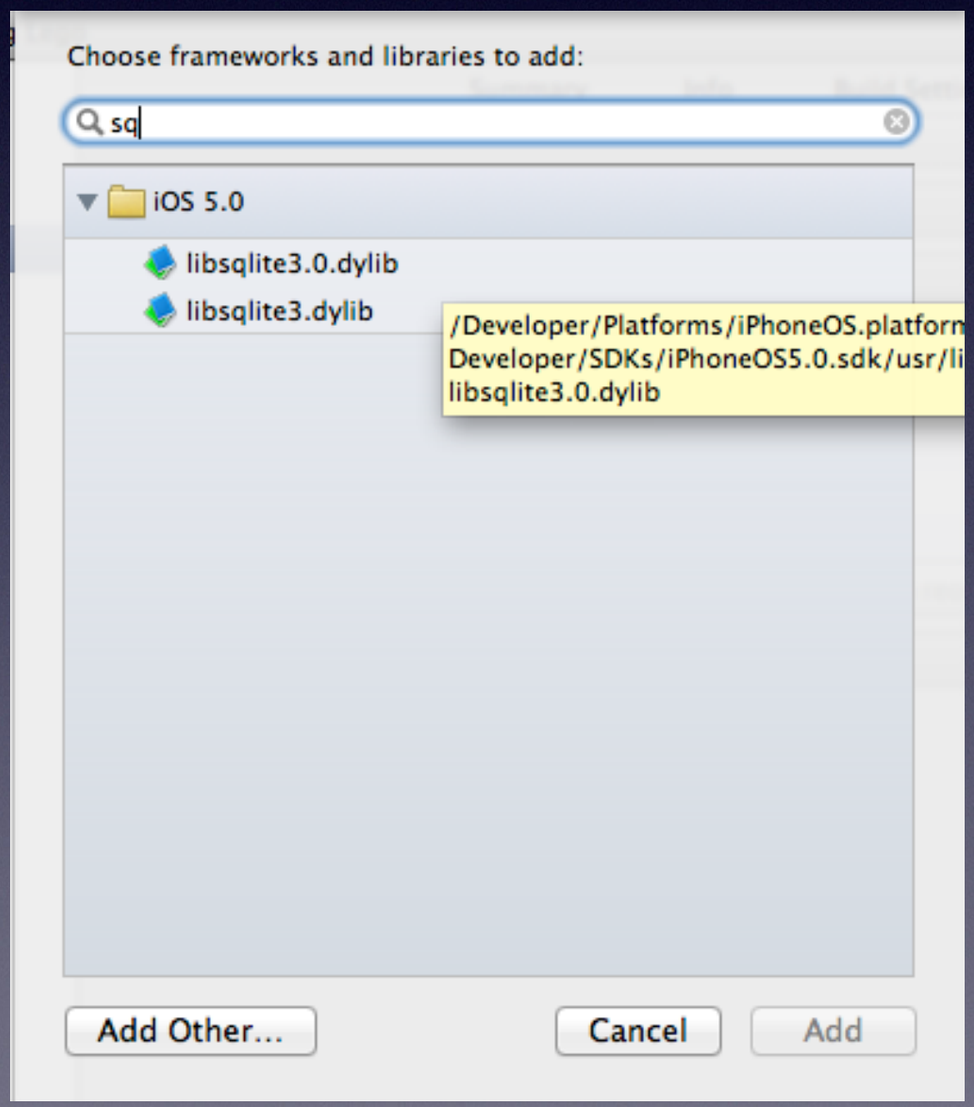
# Biblioteka SQLite

- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib



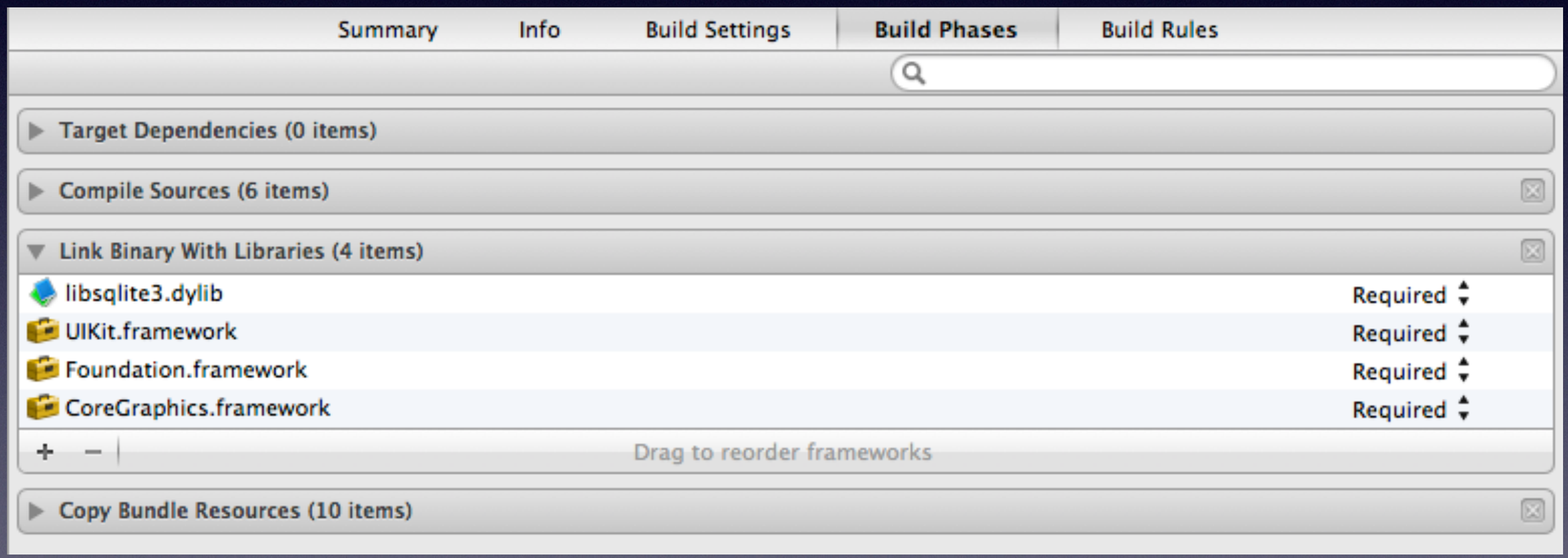
# Biblioteka SQLite

- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib



# Biblioteka SQLite

- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib



# Biblioteka SQLite

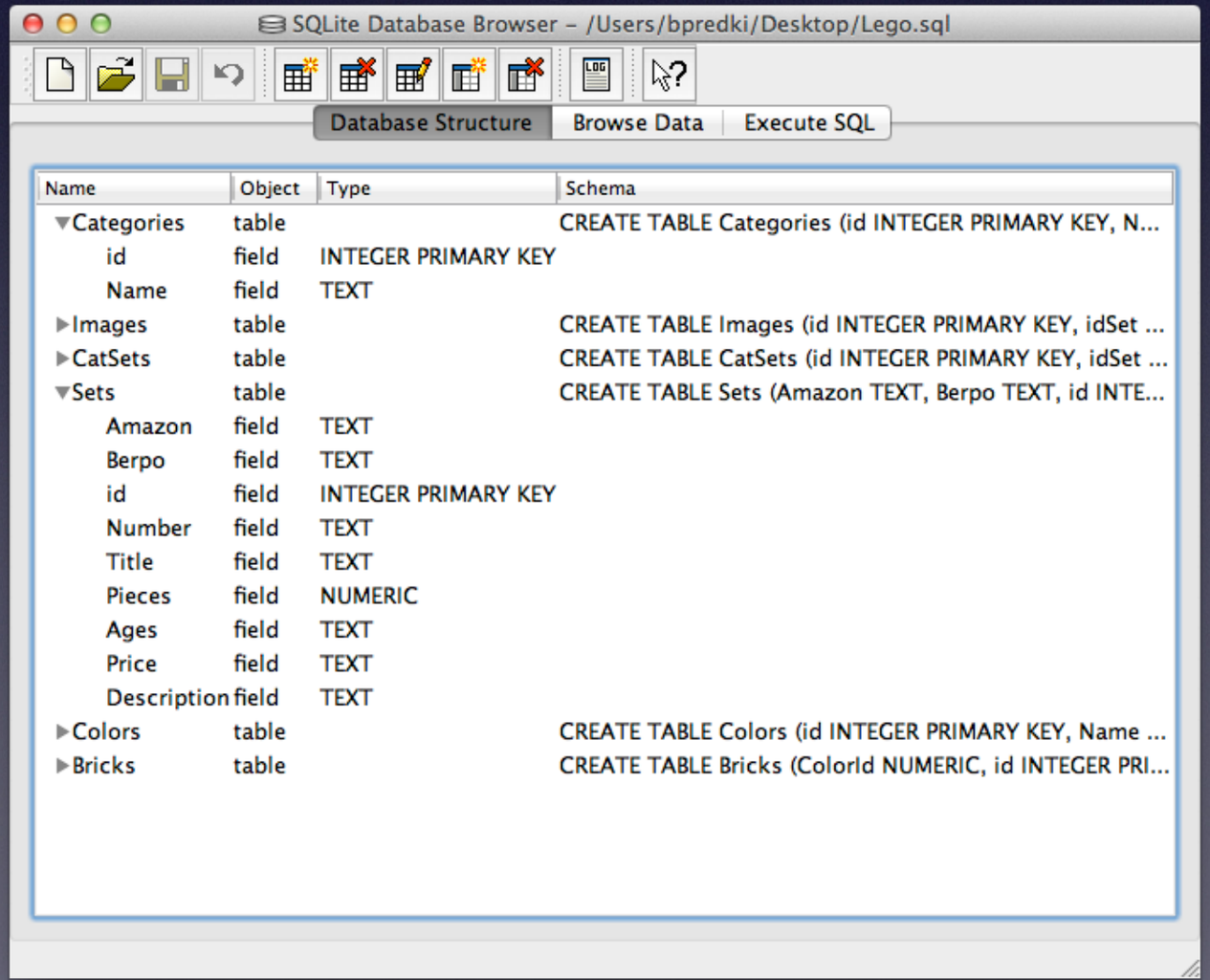
- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib

# Biblioteka SQLite

- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib
- tworzymy bazę (np. wtyczka do FireFox)

# Biblioteka SQLite

- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib
- tworzymy bazę (np. wtyczka do FireFox)



# Biblioteka SQLite

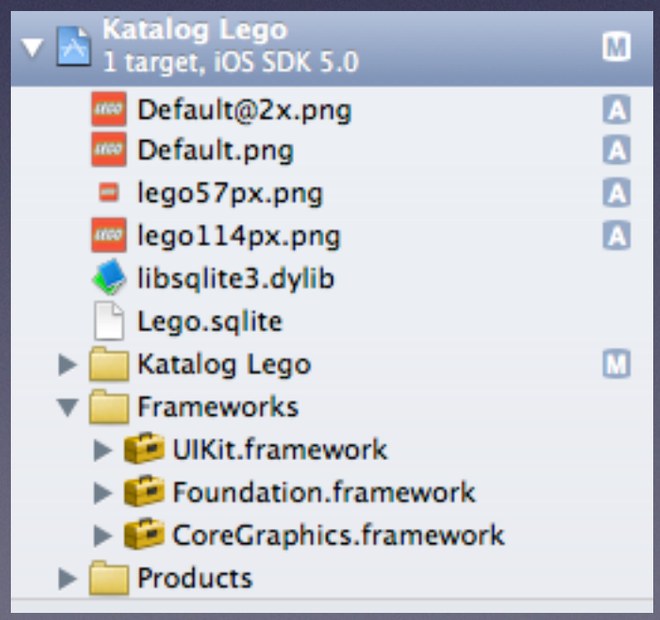
- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib
- tworzymy bazę (np. wtyczka do FireFox)

# Biblioteka SQLite

- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib
- tworzymy bazę (np. wtyczka do FireFox)
- do projektu dodajemy plik z bazą

# Biblioteka SQLite

- w zakładce Build Phases dodajemy bibliotekę libsqlite3.dylib
- tworzymy bazę (np. wtyczka do FireFox)
- do projektu dodajemy plik z bazą



# Kontroler głównego widoku

# Kontroler głównego widoku

```
//
//  RootViewController.h
//  Katalog Lego
//
//  Created by Bartłomiej Prędki on 11-06-19.
//  Copyright 2011 Telperion. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <sqlite3.h>
#include "Category2ViewController.h"

@class Category2ViewController;

@interface RootViewController : UITableViewController {
    NSString        *databasePath;
    sqlite3         *legoDB;
    NSMutableArray  *categories;
    NSMutableArray  *categoryIds;
    Category2ViewController *categoryController;
}

@property (nonatomic, retain) IBOutlet Category2ViewController *categoryController;

@end
```

# Kontroler głównego widoku

# Kontroler głównego widoku

- Przy pierwszym uruchomieniu bazę danych należy skopiować do katalogu roboczego

# Kontroler głównego widoku

- Przy pierwszym uruchomieniu bazę danych należy skopiować do katalogu roboczego
- W przyszłości aktualizacja bazy danych

# Kontroler głównego widoku

- Przy pierwszym uruchomieniu bazę danych należy skopiować do katalogu roboczego
- W przyszłości aktualizacja bazy danych

```
//
//  RootViewController.m
//  Katalog Lego
//
//  Created by Bartłomiej Prędki on 11-06-19.
//  Copyright 2011 Telperion. All rights reserved.
//

#import "RootViewController.h"

@implementation RootViewController

@synthesize categoryController;
```

# Kontroler głównego widoku

# Kontroler głównego widoku

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    NSString *docsDir;
    NSArray *dirPaths;

    // Get the documents directory
    dirPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);

    docsDir = [dirPaths objectAtIndex:0];

    // Build the path to the database file
    databasePath = [[NSString alloc] initWithString: [docsDir stringByAppendingPathComponent: @"Lego.sqlite"]];

    NSFileManager *filemgr = [NSFileManager defaultManager];

    //Jeżeli bazy nie ma to skopiuj ją z zasobów
    if ([filemgr fileExistsAtPath: databasePath ] == NO)
    {
        NSError **error;
        NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
        NSString *documentsDirectory = [paths objectAtIndex:0];
        NSString *path = [documentsDirectory stringByAppendingPathComponent:@"Lego.sqlite"];

        NSFileManager *fileManager = [NSFileManager defaultManager];

        if (![fileManager fileExistsAtPath: path])
        {
            NSString *bundle =  [[ NSBundle mainBundle] pathForResource:@"Lego" ofType:@"sqlite"];
            [fileManager copyItemAtPath:bundle toPath:path error:error];
        }
    }

    [filemgr release];
```

# Kontroler głównego widoku

# Kontroler głównego widoku

```
    //Zapamiętaj kategorie
    categories=[[NSMutableArray alloc] initWithObjects:nil];
    categoryIds=[[NSMutableArray alloc] initWithObjects:nil];

    const char *dbpath = [databasePath UTF8String];
    sqlite3_stmt    *statement;

    if (sqlite3_open(dbpath, &legoDB) == SQLITE_OK)
    {
        NSString *querySQL = [NSString stringWithFormat: @"SELECT * from Categories"];

        const char *query_stmt = [querySQL UTF8String];
        if (sqlite3_prepare_v2(legoDB, query_stmt, -1, &statement, NULL) == SQLITE_OK)
        {
            while (sqlite3_step(statement) == SQLITE_ROW)
            {
                NSString *title = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement, 1)];
                [categories addObject:title];
                NSString *catId = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement, 0)];
                [categoryIds addObject:catId];
            }
            sqlite3_finalize(statement);
        }
        sqlite3_close(legoDB);
    }
    self.title = @"Categories";
}
```

# Kontroler głównego widoku

# Kontroler głównego widoku

```objc
// Customize the number of sections in the table view.
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return [categories count];
}

// Customize the appearance of table view cells.
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier] autorelease];
    }

    // Configure the cell.
    cell.textLabel.text = [categories objectAtIndex: [indexPath row]];

    return cell;
}
```

# Kontroler głównego widoku

# Kontroler głównego widoku

```
- (void)dealloc
{
    [categories release];
    [categoryIds release];
    [super dealloc];
}

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    self.categoryController.title = [categories objectAtIndex: [indexPath row]];
    self.categoryController.catId = [categoryIds objectAtIndex: [indexPath row]];
    self.categoryController.databasePath=databasePath;

    [self.navigationController
     pushViewController:self.categoryController
     animated:YES];
}
```

# Kontroler widoku kategorii

# Kontroler widoku kategorii

```
//
//  Category2ViewController.h
//  Katalog Lego
//
//  Created by Bartłomiej Prędki on 11-06-21.
//  Copyright 2011 Telperion. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <sqlite3.h>
#include "SetViewController.h"

@interface Category2ViewController : UITableViewController {
    NSMutableArray *sets;
    NSString *title;
    NSString *catId;
    NSString *databasePath;
    sqlite3  *legoDB;
    NSMutableArray *setscollection;
    SetViewController *detailViewController;
}

@property (nonatomic, retain) NSMutableArray *sets;
@property (nonatomic, retain) NSString *title;
@property (nonatomic, retain) NSString *catId;
@property (nonatomic, retain) NSString *databasePath;
@property (nonatomic,retain) IBOutlet SetViewController *detailViewController;

@end
```

# Kontroler widoku kategorii

# Kontroler widoku kategorii

```
//
//  Category2ViewController.m
//  Katalog Lego
//
//  Created by Bartłomiej Prędki on 11-06-21.
//  Copyright 2011 Telperion. All rights reserved.
//

#import "Category2ViewController.h"


@implementation Category2ViewController

@synthesize sets,title,catId,databasePath,detailViewController;

- (id)initWithStyle:(UITableViewStyle)style
{
    self = [super initWithStyle:style];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)dealloc
{
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}
```

# Kontroler widoku kategorii

# Kontroler widoku kategorii

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    // Uncomment the following line to preserve selection between presentations.
    // self.clearsSelectionOnViewWillAppear = NO;

    // Uncomment the following line to display an Edit button in the navigation bar for this view controller.
    // self.navigationItem.rightBarButtonItem = self.editButtonItem;
    SetViewController *controller = [[SetViewController alloc] initWithNibName:@"SetViewController" bundle:nil];
    self.detailViewController=controller;

}

- (void)viewDidUnload
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
    [self.detailViewController release];
}
```

```objc
- (void)viewWillAppear:(BOOL)animated
{
    NSMutableArray *catsets=[[NSMutableArray alloc] initWithObjects:nil];
    setscollection=[[NSMutableArray alloc] initWithObjects:nil];


    const char *dbpath = [databasePath UTF8String];
    sqlite3_stmt    *statement;

    if (sqlite3_open(dbpath, &legoDB) == SQLITE_OK)
    {
        NSString *querySQL = [NSString stringWithFormat: @"SELECT idSet from CatSets where idCat=%@",catId];

        const char *query_stmt = [querySQL UTF8String];
        if (sqlite3_prepare_v2(legoDB, query_stmt, -1, &statement, NULL) == SQLITE_OK)
        {
            while (sqlite3_step(statement) == SQLITE_ROW)
            {
                NSString *setId = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement, 0)];
                [catsets addObject:setId];
            }
            sqlite3_finalize(statement);
        }

        for(int n = 0; n < [catsets count] ; n = n + 1)
        {
            querySQL=[NSString stringWithFormat: @"SELECT id,number,title,pieces,ages, price,description,berpo,amazon from Sets where id=%@",
                [catsets objectAtIndex:n]];
            const char *query_stmt2 = [querySQL UTF8String];
            if (sqlite3_prepare_v2(legoDB, query_stmt2, -1, &statement, NULL) == SQLITE_OK)
            {
                if (sqlite3_step(statement) == SQLITE_ROW)
                {
                    LegoSet *set=[[LegoSet alloc] initWithStatement:statement];
                    [setscollection addObject:set];
                }
                sqlite3_finalize(statement);
            }

        }


        sqlite3_close(legoDB);
    }

    [self.tableView reloadData];
    [super viewWillAppear:animated];
}
```

# Kontroler widoku kategorii

# Kontroler widoku kategorii

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    return [setscollection count];
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";
    UITableViewCell *cell = [tableView
                             dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc]
                 initWithStyle:UITableViewCellStyleDefault
                 reuseIdentifier:CellIdentifier] autorelease];
    }
    // Configure the cell…
    LegoSet *set=[setscollection objectAtIndex: [indexPath row]];

    cell.textLabel.text = [[NSString alloc] initWithString:[set formatTitle]];
    return cell;

}
```

# Kontroler widoku kategorii

# Kontroler widoku kategorii

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    // Navigation logic may go here. Create and push another view controller.

    LegoSet *set=[setscollection objectAtIndex: [indexPath row]];
    self.detailViewController.set=set;

    [self.navigationController
     pushViewController:self.detailViewController
     animated:YES];


}
```

# Klasa zestawu

# Klasa zestawu

```
//
//  LegoSet.h
//  Katalog Lego
//
//  Created by Bartłomiej Prędki on 11-06-20.
//  Copyright 2011 Telperion. All rights reserved.
//

#import <Foundation/Foundation.h>
#import <sqlite3.h>


@interface LegoSet : NSObject {
    NSInteger idSet;
    NSString *number;
    NSString *title;
    NSString *ages;
    NSString *price;
    NSString *berpo;
    NSString *amazon;
    NSString *description;
    NSInteger pieces;
    UIImage *image;

    NSString        *databasePath;
    sqlite3         *legoDB;
}

@property (nonatomic, retain) NSString *number;
@property (nonatomic, retain) NSString *title;
@property (nonatomic, retain) NSString *ages;
@property (nonatomic, retain) NSString *price;
@property (nonatomic, retain) NSString *berpo;
@property (nonatomic, retain) NSString *amazon;
@property (nonatomic, retain) NSString *description;
@property (nonatomic) NSInteger pieces;
@property (nonatomic) NSInteger idSet;
@property (nonatomic, retain) UIImage *image;

-(LegoSet*) initWithStatement: (sqlite3_stmt*) statement;
-(void) loadImage;
-(NSString*) formatTitle;

@end
```

# Klasa zestawu

# Klasa zestawu

```
//
//  LegoSet.m
//  Katalog Lego
//
//  Created by Bartłomiej Prędki on 11-06-20.
//  Copyright 2011 Telperion. All rights reserved.
//

#import "LegoSet.h"


@implementation LegoSet

@synthesize number,title,price,ages,description,pieces,image,berpo,amazon,idSet;

-(LegoSet*) initWithStatement: (sqlite3_stmt*) statement
{
    self = [super init];

    if (self) {
        idSet = sqlite3_column_int(statement, 0);
        number = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement, 1)];
        title = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement, 2)];
        pieces = sqlite3_column_int(statement, 3);
        ages = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement, 4)];
        price = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement, 5)];
        description=[[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement, 6)];
        char *value = sqlite3_column_text(statement, 7);

        if (value == NULL)
            self.berpo = nil;
        else
            self.berpo = [NSString stringWithUTF8String: value];

        amazon= nil;//[[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement, 8)];

        [self loadImage];
    }

    return self;
}
```

# Klasa zestawu

# Klasa zestawu

```
-(void) loadImage
{
    NSString *docsDir;
    NSArray *dirPaths;

    // Get the documents directory
    dirPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);

    docsDir = [dirPaths objectAtIndex:0];

    // Build the path to the database file
    databasePath = [[NSString alloc] initWithString: [docsDir stringByAppendingPathComponent: @"Lego.sqlite"]];
    const char *dbpath = [databasePath UTF8String];
    sqlite3_stmt    *statement;

    if (sqlite3_open(dbpath, &legoDB) == SQLITE_OK)
    {
        NSString *querySQL = [NSString stringWithFormat: @"SELECT * from Images where idSet=%d",idSet];

        const char *query_stmt = [querySQL UTF8String];
        if (sqlite3_prepare_v2(legoDB, query_stmt, -1, &statement, NULL) == SQLITE_OK)
        {
            while (sqlite3_step(statement) == SQLITE_ROW)
            {
                NSData *data = [[NSData alloc] initWithBytes:sqlite3_column_blob(statement, 2) length:sqlite3_column_bytes(statement, 2)];
                if(data == nil)
                    self.image=nil;
                else
                    self.image = [UIImage imageWithData:data];
            }
            sqlite3_finalize(statement);
        }
        sqlite3_close(legoDB);
    }

}

-(NSString*) formatTitle
{
    NSString* t=[[NSString alloc] initWithFormat:@"%@-%@",number,title];
    return t;
}
```

# Kontroler widoku zestawu

# Kontroler widoku zestawu

```
//
//  SetViewController.h
//  Katalog Lego
//
//  Created by Bartłomiej Prędki on 11-06-22.
//  Copyright 2011 Telperion. All rights reserved.
//

#import <UIKit/UIKit.h>
#include "LegoSet.h"


@interface SetViewController : UIViewController {

    IBOutlet UIImageView *image;
    IBOutlet UILabel *setTitle;
    IBOutlet UILabel *pieces;
    IBOutlet UILabel *ages;
    IBOutlet UILabel *price;
    IBOutlet UILabel *berpo;
    IBOutlet UIWebView *description;

    LegoSet *set;
}

@property (nonatomic,retain) LegoSet *set;
@property (nonatomic,retain) IBOutlet UILabel *setTitle;
@property (nonatomic,retain) IBOutlet UILabel *pieces;
@property (nonatomic,retain) IBOutlet UILabel *ages;
@property (nonatomic,retain) IBOutlet UILabel *price;
@property (nonatomic,retain) IBOutlet UILabel *berpo;
@property (nonatomic,retain) IBOutlet UIWebView *description;
@property (nonatomic,retain) IBOutlet UIImageView *image;

-(void) Refresh;

@end
```

# Kontroler widoku zestawu

# Kontroler widoku zestawu

```
//
//  SetViewController.m
//  Katalog Lego
//
//  Created by Bartłomiej Prędki on 11-06-22.
//  Copyright 2011 Telperion. All rights reserved.
//

#import "SetViewController.h"


@implementation SetViewController

@synthesize set,setTitle,price,ages,image,pieces,description,berpo;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)dealloc
{
    [image release];
    [setTitle release];
    [pieces release];
    [ages release];
    [price release];
    [description release];
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}
```

```objc
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view from its nib.
    //[self Refresh];
}

- (void)viewDidUnload
{
    [image release];
    image = nil;
    [setTitle release];
    setTitle = nil;
    [pieces release];
    pieces = nil;
    [ages release];
    ages = nil;
    [price release];
    price = nil;
    [description release];
    description = nil;
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

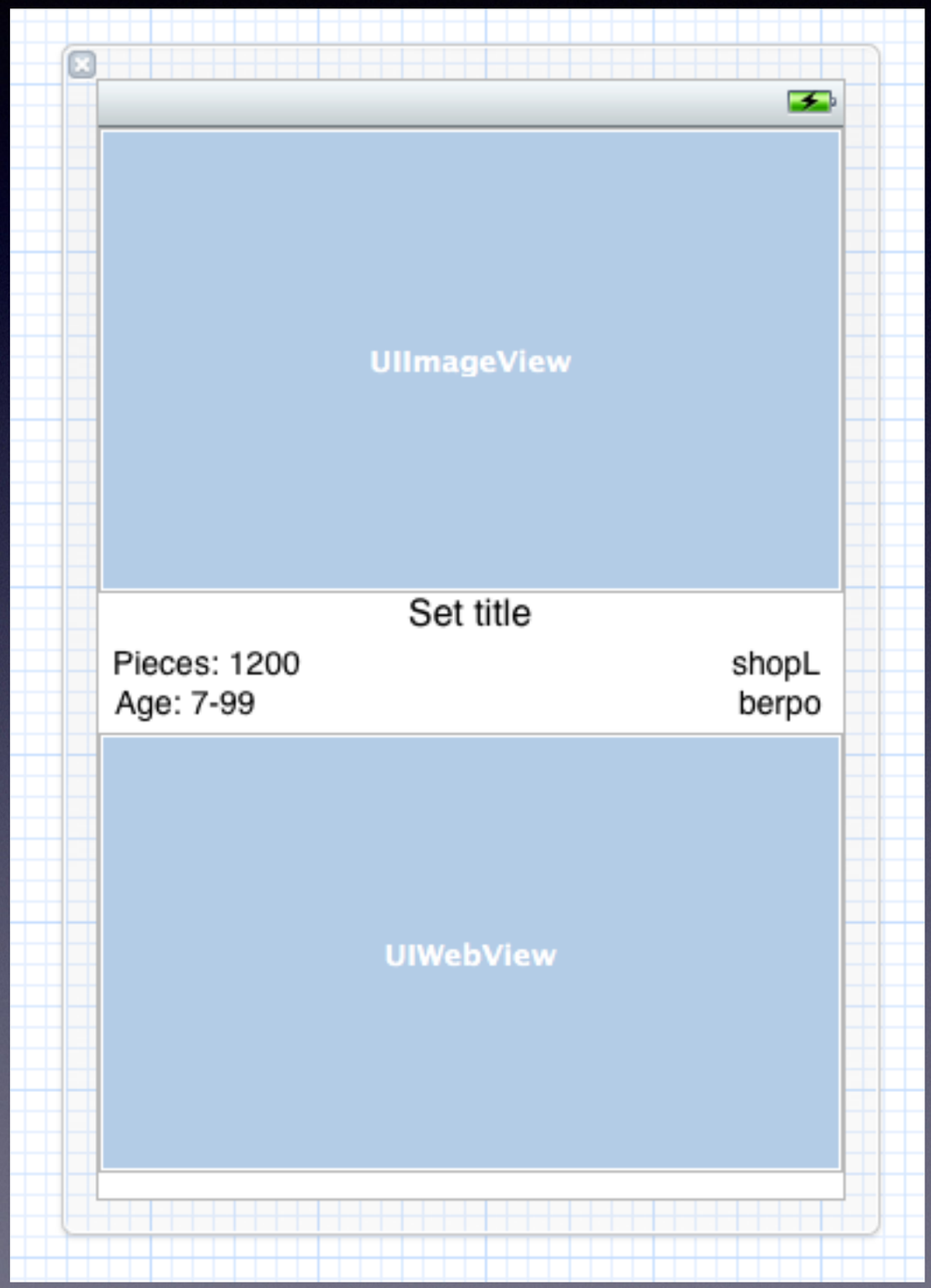
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

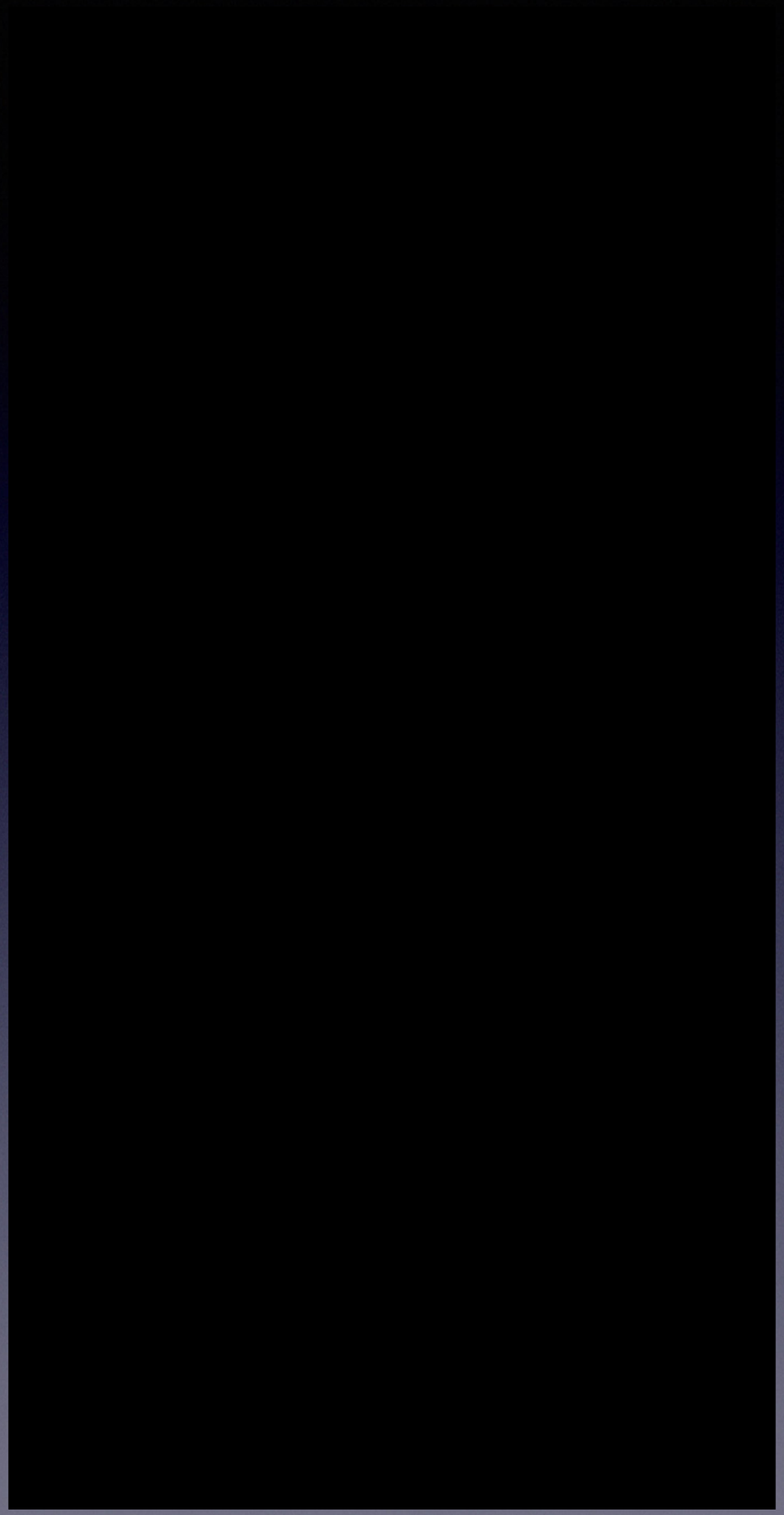
-(void)viewWillAppear:(BOOL)animated
{
    [self Refresh];
}

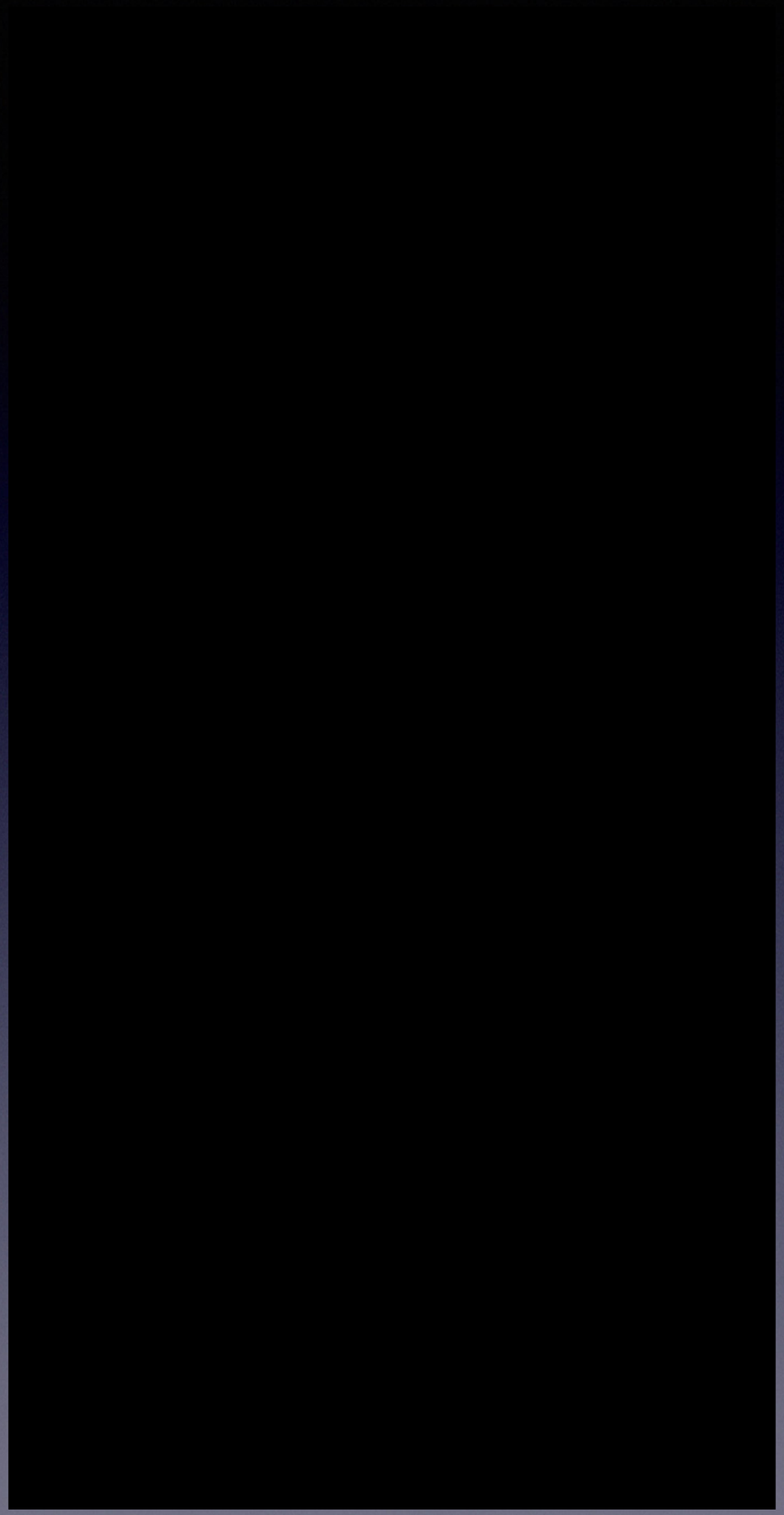
-(void) Refresh
{
    [setTitle setText: [set title]];
    [pieces setText: [[NSString alloc] initWithFormat:@"Pieces: %d",[set pieces]]];
    [ages setText:[[NSString alloc] initWithFormat:@"Ages: %@",[set ages]]];
    [price setText:[[NSString alloc] initWithFormat:@"%@ zł",[set price]]];
    [berpo setText:[[NSString alloc] initWithFormat:@"%@ zł",[set berpo]]];
    [description loadHTMLString:[set description] baseURL:[NSURL fileURLWithPath:[[NSBundle mainBundle] bundlePath]]];
    self.title=[set number];
    self.image.image = [set image];
}
```

# Widok zestawu

# Widok zestawu







# Pickery

# Pickery

# Pickery

- Odpowiedniki ComboBox'ów

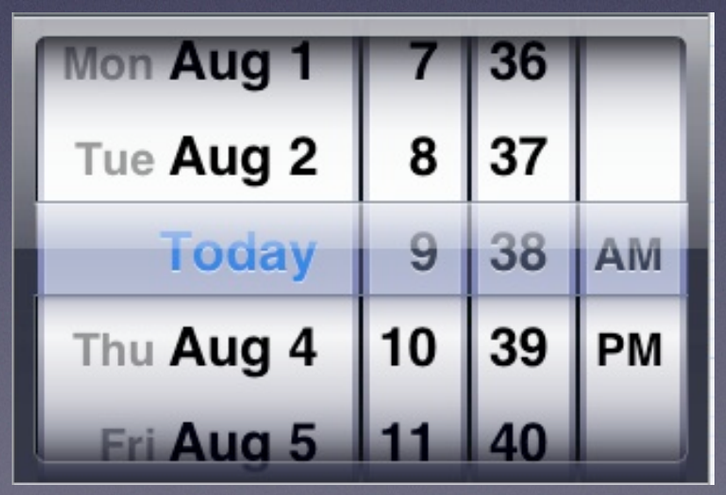
# Pickery

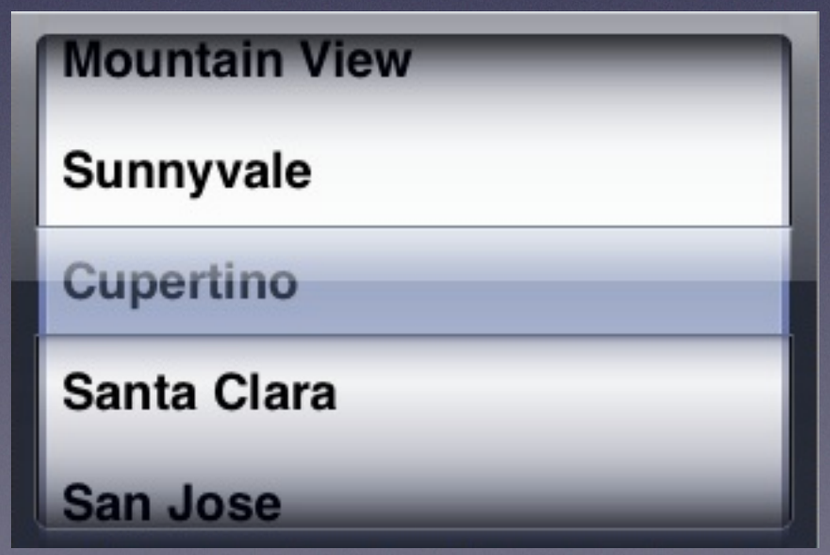
- Odpowiedniki ComboBox'ów
- Specjalizowany DatePicker



# Pickery

- Odpowiedniki ComboBox'ów
- Specjalizowany DatePicker
- Ogólny UIPickerView



# Przykład

# Przykład

- Definiujemy gniazdka i wtyczki

# Przykład

- Definiujemy gniazdka i wtyczki

```
//
//  ViewController.h
//  Pickers
//
//  Created by Bartlomiej Predki on 15.05.2012.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController {
    UIDatePicker *data;
    UILabel *etykietka;
}

@property (strong,nonatomic) IBOutlet UIDatePicker *data;
@property (strong,nonatomic) IBOutlet UILabel *etykietka;
-(IBAction)pobierzDate;

@end
```

# Przykład

- Definiujemy gniazdka i wtyczki

```
//
//  ViewController.h
//  Pickers
//
//  Created by Bartlomiej Predki on 15.05.2012.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController {
    UIDatePicker *data;
    UILabel *etykietka;
}

@property (strong,nonatomic) IBOutlet UIDatePicker *data;
@property (strong,nonatomic) IBOutlet UILabel *etykietka;
-(IBAction)pobierzDate;

@end
```

```
#import "ViewController.h"

@implementation ViewController
@synthesize data,etykietka;
```

# Przykład

- Definiujemy gniazdka i wtyczki
- Projektujemy widok

```
//
//  ViewController.h
//  Pickers
//
//  Created by Bartlomiej Predki on 15.05.2012.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController {
    UIDatePicker *data;
    UILabel *etykietka;
}

@property (strong,nonatomic) IBOutlet UIDatePicker *data;
@property (strong,nonatomic) IBOutlet UILabel *etykietka;
-(IBAction)pobierzDate;

@end
```

```
#import "ViewController.h"

@implementation ViewController
@synthesize data,etykietka;
```

# Przykład

- Definiujemy gniazdka i wtyczki
- Projektujemy widok

```
//
//  ViewController.h
//  Pickers
//
//  Created by Bartlomiej Predki on 15.05.2012.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

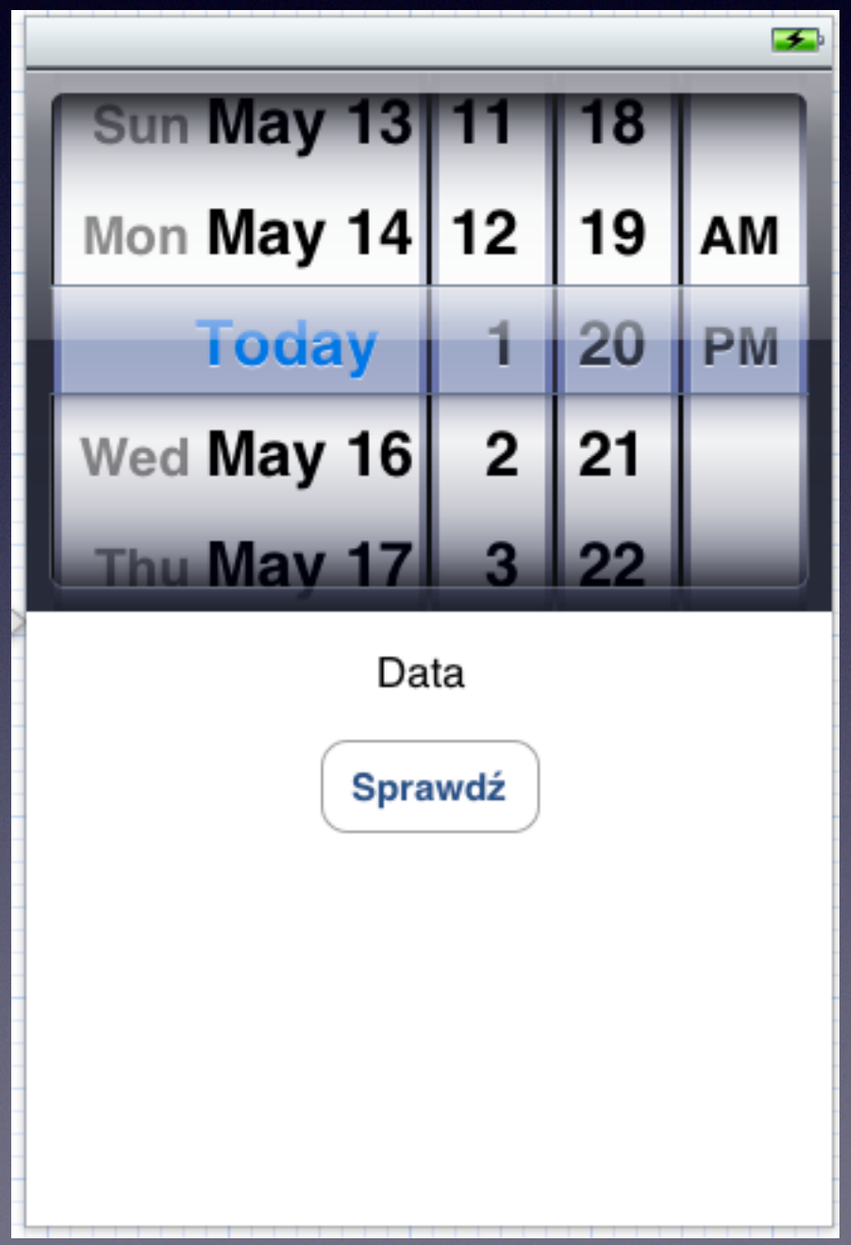
@interface ViewController : UIViewController {
    UIDatePicker *data;
    UILabel *etykietka;
}

@property (strong,nonatomic) IBOutlet UIDatePicker *data;
@property (strong,nonatomic) IBOutlet UILabel *etykietka;
-(IBAction)pobierzDate;

@end
```

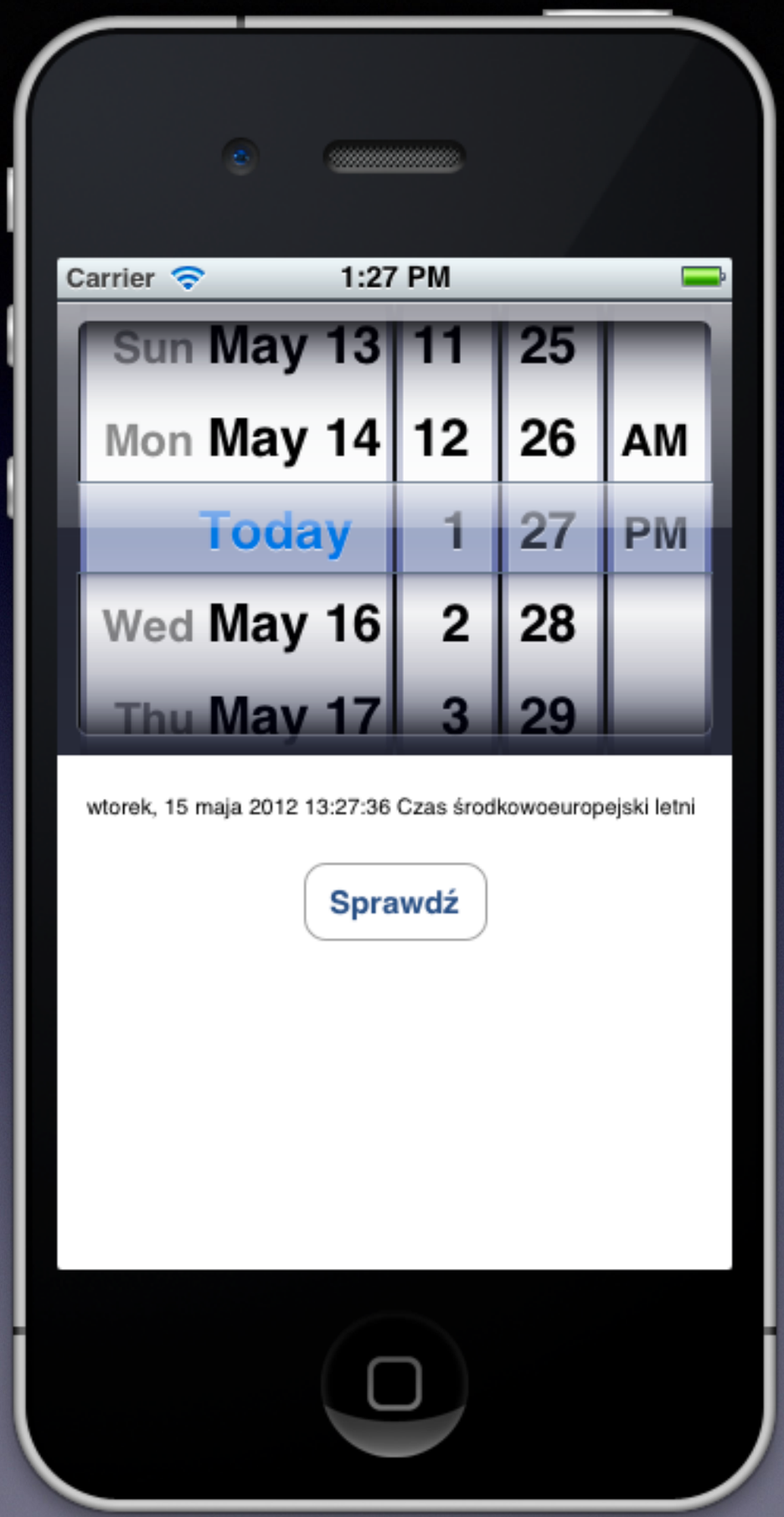
```
#import "ViewController.h"

@implementation ViewController
@synthesize data,etykietka;
```



# Efekt

# Efekt


Carrier
1:27 PM
Sun May 13 11 25
Mon May 14 12 26 AM
Today 1 27 PM
Wed May 16 2 28
Thu May 17 3 29
wtorek, 15 maja 2012 13:27:36 Czas środkowoeuropejski letni
Sprawdź

# UIPickerView

# UIPickerView

- Uniwersalny picker

# UIPickerView

- Uniwersalny picker
- Wymaga podania klasy stanowiącej źródło danych

# UIPickerView

- Uniwersalny picker
- Wymaga podania klasy stanowiącej źródło danych
- Klasa musi realizować protokół UIPickerViewDataSource

# UIPickerView

- Uniwersalny picker
- Wymaga podania klasy stanowiącej źródło danych
- Klasa musi realizować protokół UIPickerViewDataSource
- Najlepiej gry realizuje go kontroler widoku

# UIPickerViewDataSource

# UIPickerViewDataSource

- Musi posiadać metody:

# UIPickerViewDataSource

- Musi posiadać metody:
  - numberOfComponentsInPickerView

# UIPickerViewDataSource

- Musi posiadać metody:
  - numberOfComponentsInPickerView
  - numberOfRowsInComponent

# UIPickerViewDataSource

- Musi posiadać metody:
  - numberOfComponentsInPickerView
  - numberOfRowsInComponent
  - titleForRow

# UIPickerViewDelegate

# UIPickerViewDelegate

- Protokół obsługujący zdarzenia związane z działaniem Pickera

# UIPickerViewDelegate

- Protokół obsługujący zdarzenia związane z działaniem Pickera
- Najważniejsza metoda:

# UIPickerViewDelegate

- Protokół obsługujący zdarzenia związane z działaniem Pickera
- Najważniejsza metoda:
  - didSelectRow

# Przykład

# Przykład

- Konwerter między walutami

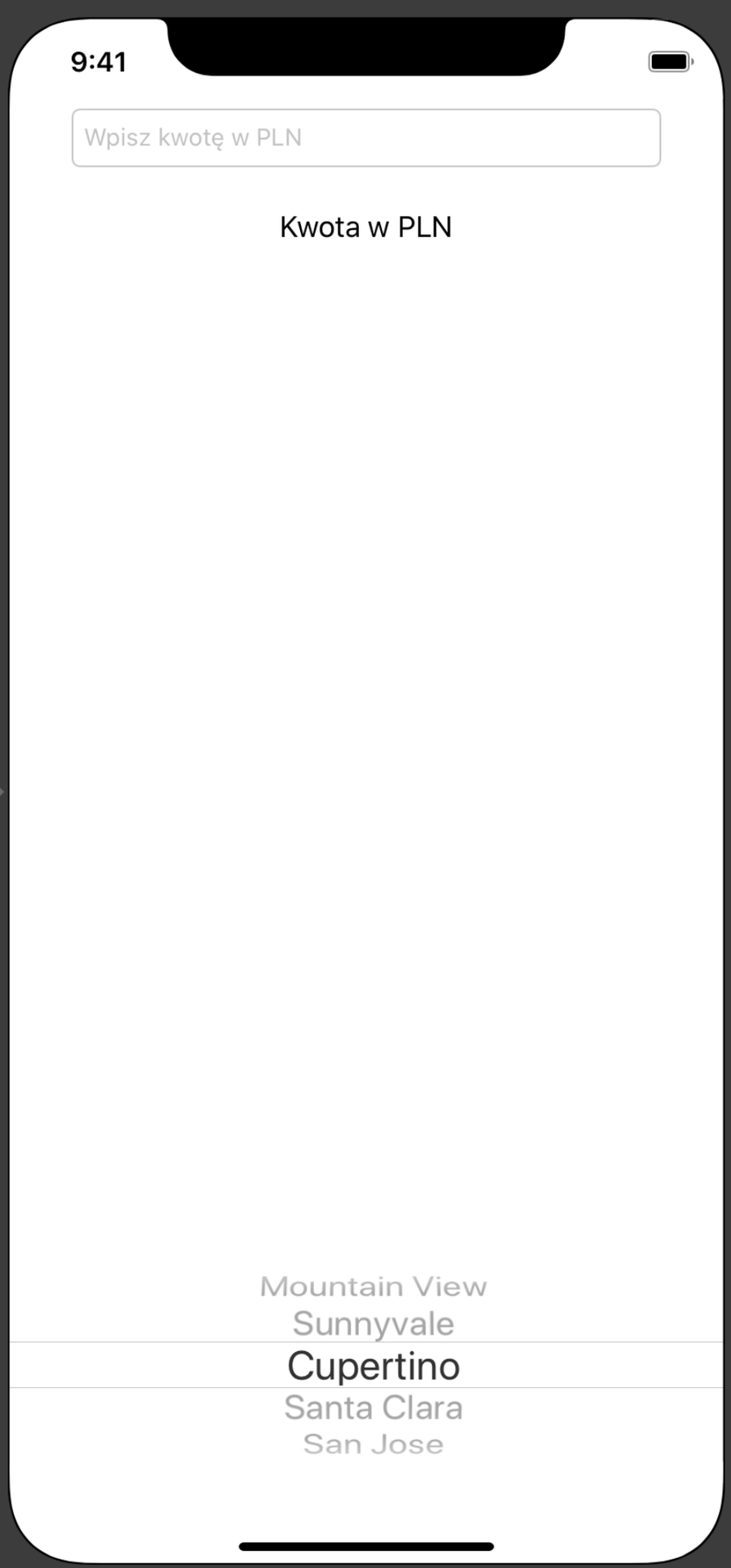
# Przykład

- Konwerter między walutami
- Tworzymy nowy projekt Single View

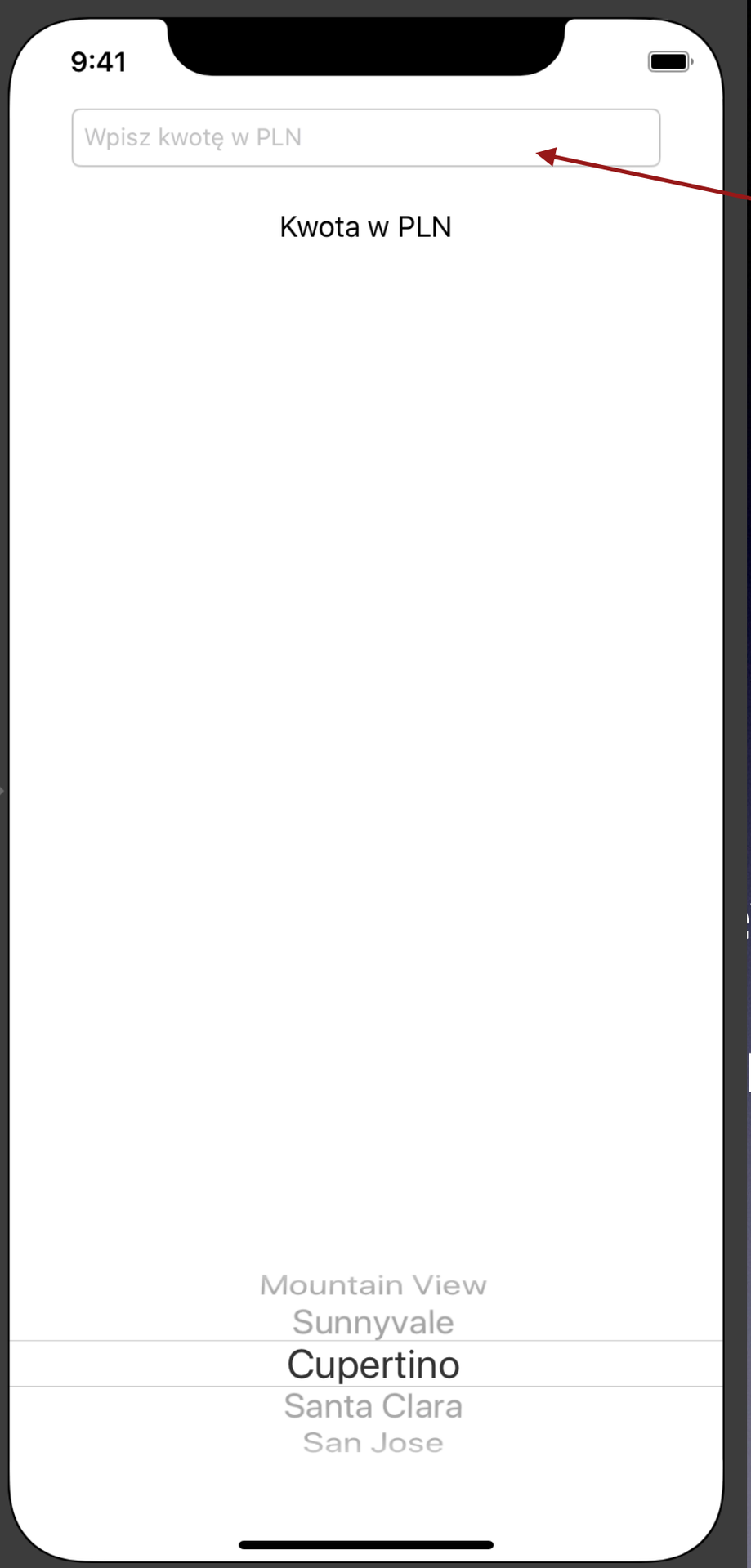
# Przykład

- Konwerter między walutami
- Tworzymy nowy projekt Single View
- Projektujemy interfejs w storyboardzie

- Konwe
- Tworzy w
- Projekt rdzie

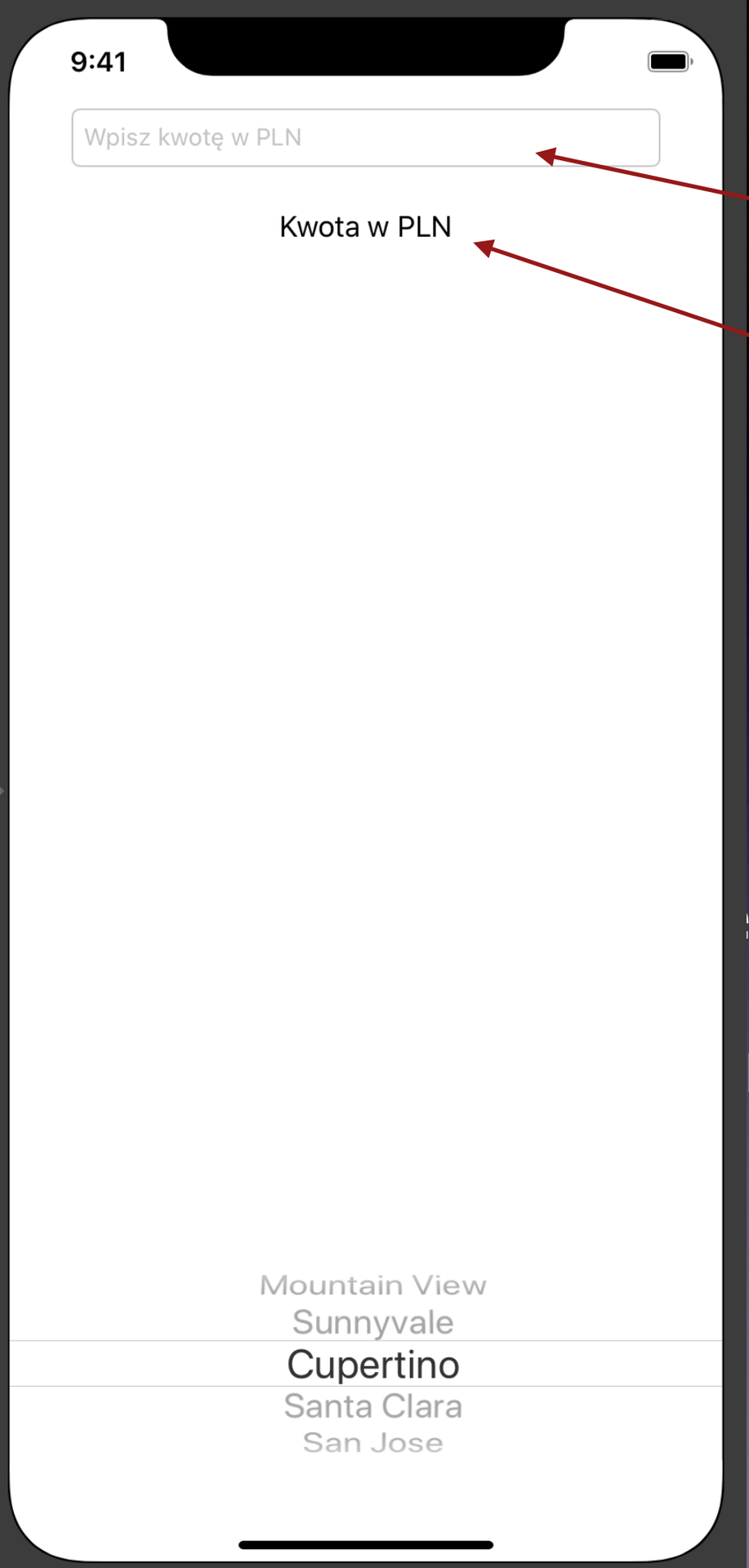


- Konwe
- Tworzy ew
- Projekt rdzie




TextField

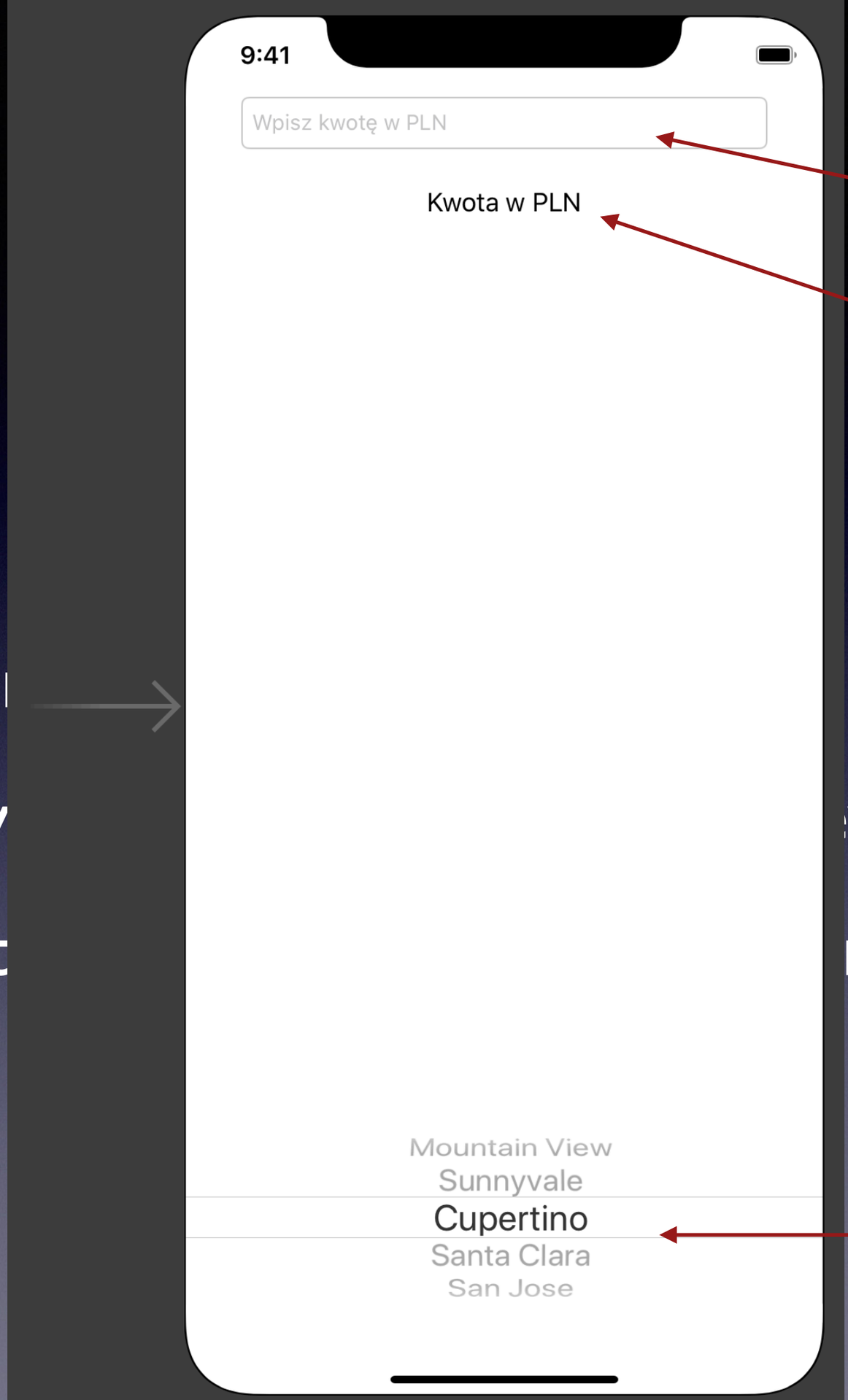
- Konwe
- Tworzy ew
- Projekt rdzie




TextField

Label

- Konwe
- Tworzy ew
- Projekt rdzie




TextField

Label

Picker

# Kursy walut

# Kursy walut

- Kursy pobierane są przy starcie z serwera jako JSON

# Kursy walut

- Kursy pobierane są przy starcie z serwera jako JSON


```
{
        "USD-PLN":"4.248",
        "USD-EUR":"0.9268",
        "USD-CHF":"0.974",
        "USD-GBP":"0.820",
        "EUR-PLN":"4.581",
        "EUR-CHF":"1.051",
        "EUR-GBP":"0.884",
        "GBP-PLN":"5.1807",
        "GBP-CHF":"1.189",
        "CHF-PLN":"4.3588"
}
```

# Dostęp do domeny

# Dostęp do domeny

- W pliku Info.plist musimy ustawić dozwoloną domenę, z którą aplikacja może się łączyć

# Dostęp do domeny

- W pliku Info.plist musimy ustawić dozwoloną domenę, z którą aplikacja może się łączyć

| | | |
|---|---|---|
| ▼ Information Property List | Dictionary | (16 items) |
| ▼ App Transport Security Settings | Dictionary | (2 items) |
| ▼ Exception Domains | Dictionary | (1 item) |
| ▼ fcds.cs.put.poznan.pl | Dictionary | (3 items) |
| NSIncludesSubdomains | Boolean | YES |
| NSTemporaryExceptionAllow... | Boolean | YES |
| NSTemporaryExceptionMinim... | String | TLSv1.1 |
| Allow Arbitrary Loads | Boolean | NO |

# Kod klasy kontrolera

# Kod klasy kontrolera

- Najpierw łączymy gniazdka i akcje:

# Kod klasy kontrolera

- Najpierw łączymy gniazdka i akcje:
  - G - do pola tekstowego

# Kod klasy kontrolera

- Najpierw łączymy gniazdka i akcje:
  - G - do pola tekstowego
  - G - do etykietki

# Kod klasy kontrolera

- Najpierw łączymy gniazdka i akcje:
  - G - do pola tekstowego
  - G - do etykietki
  - G - do pickera

# Kod klasy kontrolera

- Najpierw łączymy gniazdka i akcje:
  - G - do pola tekstowego
  - G - do etykietki
  - G - do pickera
  - A - do pola tekstowego (EditingChanged)

# Kod klasy kontrolera

- Najpierw łączymy gniazdka i akcje:
  - G - do pola tekstowego
  - G - do etykietki
  - G - do pickera
  - A - do pola tekstowego (EditingChanged)
  - Łączymy też Delegate pickera z kontrolerem widoku

# Kod klasy kontrolera

- Najpierw łączymy gniazdka i akcje:
  - G - do pola tekstowego
  - G - do etykietki
  - G - do pickera

```
@IBAction func editingChanged(_ sender: Any) {
    convertValue()
}

@IBOutlet weak var textInput: UITextField!
@IBOutlet weak var labelOutput: UILabel!
@IBOutlet weak var pickerCurrencies: UIPickerView!
```

ed)

# Klasa kontrolera

# Klasa kontrolera

- Dodajemy do klasy 2 protokoły:

# Klasa kontrolera

- Dodajemy do klasy 2 protokoły:
  - UIPickerViewDataSource

# Klasa kontrolera

- Dodajemy do klasy 2 protokoły:
  - UIPickerViewDataSource
  - UIPickerViewDelegate

# Klasa kontrolera

- Dodajemy do klasy 2 protokoły:
  - UIPickerViewDataSource
  - UIPickerViewDelegate

```
class ViewController: UIViewController, UIPickerViewDataSource, UIPickerViewDelegate {
```

# Klasa kontrolera

# Klasa kontrolera

- Dodajemy pola do przechowywania danych

# Klasa kontrolera

- Dodajemy pola do przechowywania danych

```
private let currencies = ["PLN","EUR","USD","GBP","CHF"]
private var rates = [String:Double]()
private var from=0, to=0
```

# Klasa kontrolera

# Klasa kontrolera

- Oraz definiujemy ich metody:

# Klasa kontrolera

- Oraz definiujemy ich metody:
  - numberOfComponents

# Klasa kontrolera

- Oraz definiujemy ich metody:
  - numberOfComponents

```
func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 2
}
```

# Klasa kontrolera

- Oraz definiujemy ich metody:
  - numberOfComponents

```
func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 2
}
```

  - numberOfRowsInComponent

# Klasa kontrolera

- Oraz definiujemy ich metody:
  - numberOfComponents

```
func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 2
}
```

  - numberOfRowsInComponent

```
func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
    switch component {
    case 0:
        return currencies.count
    case 1:
        return currencies.count
    default:
        return 0
    }
}
```

# Klasa kontrolera

# Klasa kontrolera

- Oraz
  - titleForRow

# Klasa kontrolera

- Oraz
  - titleForRow

```
func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? {
    return currencies[row]
}
```

# Klasa kontrolera

# Klasa kontrolera

- I w końcu metodę uruchamianą, gdy użytkownik zmieni wartość pickera

# Klasa kontrolera

- I w końcu metodę uruchamianą, gdy użytkownik zmieni wartość pickera

```
func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
    switch component {
    case 0:
        from = row
        textInput.placeholder = "Wprowadź kwotę w \(currencies[row])"
    case 1:
        to = row
        labelOutput.text = "Kwota w \(currencies[row])"
    default:
        return
    }
    convertValue()
}
```

# Klasa kontrolera

# Klasa kontrolera

- Dodamy teraz pobieranie kursów w metodzie viewDidLoad

# Klasa kontrolera

- Dodamy teraz pobieranie kursów w metodzie viewDidLoad

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.
    fetchData { (dict, error) in
        for k in dict!.keys {
            let rate = Double(dict![k]!)
            self.rates[k]=rate
        }
    }
}
```

# Klasa kotrolera

# Klasa kotrolera

- Faktyczne pobieranie

# Klasa kotrolera

- Faktyczne pobieranie

```
func fetchData(completion: @escaping ([String:String]?, Error?) -> Void) {
    let url = URL(string: "http://fcds.cs.put.poznan.pl/MyWeb/Media/currencies.json")!

    let task = URLSession.shared.dataTask(with: url) { (data, response, error) in
        guard let data = data else { return }
        do {
            if let array = try JSONSerialization.jsonObject(with: data, options: .allowFragments) as? [String:String]{
                completion(array, nil)
            }
        } catch {
            print(error)
            completion(nil, error)
        }
    }
    task.resume()
}
```

# Klasa kontrolera

# Klasa kontrolera

- Oraz przeliczanie walut

# Klasa kontrolera

- Oraz przeliczanie walut

```
private func convertValue() {
    var rate = 1.0
    var key = "\(currencies[from])-\(currencies[to])"
    if rates.keys.contains(key) {
        rate = rates[key]!
    } else {
        key = "\(currencies[to])-\(currencies[from])"
        if rates.keys.contains(key) {
            rate = 1/rates[key]!
        }
    }

    if let input = textInput.text {
        if let value = Double(input) {
            let result = value * rate
            labelOutput.text = "\(String(format: "%.2f", result)) \(currencies[to])"
        } else {
            labelOutput.text = "Wprowadź prawidłową kwotę"
        }
    } else {
        labelOutput.text = "Wprowadź prawidłową kwotę"
    }
}
```

# Klasa kontrolera

# Klasa kontrolera

- A na koniec chowanie klawiatury

# Klasa kontrolera

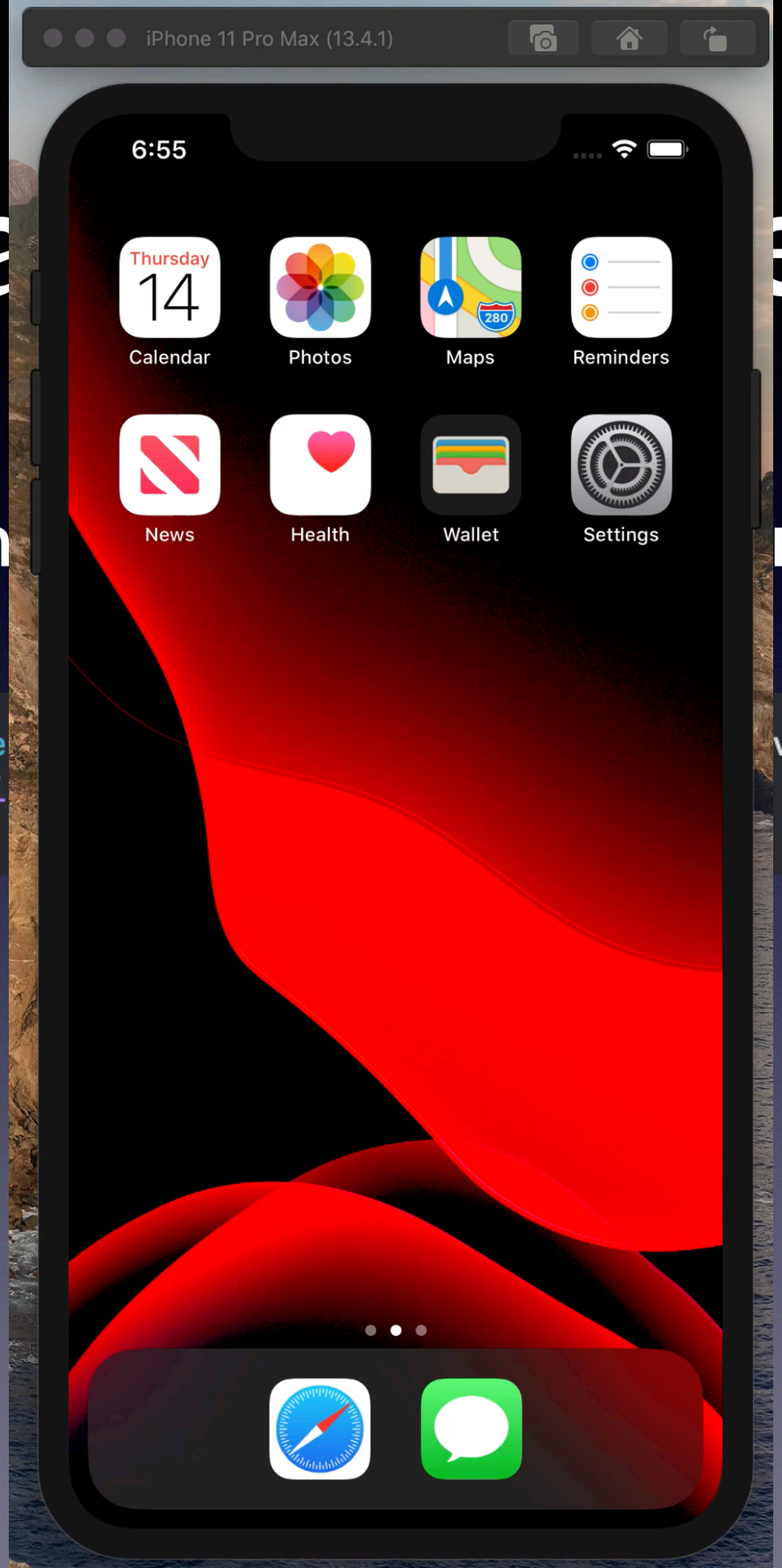
- A na koniec chowanie klawiatury

```
override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
    textInput.endEditing(true)
}
```

# Kla era

- A na kon ry

```
override func touche                                    vent: UIEvent?) {
    textInput.endEdi
}
```



# TouchID

# TouchID

# TouchID

- Wykorzystywane do potwierdzania tożsamości za pomocą czytnika linii papilarnych

# TouchID

- Wykorzystywane do potwierdzania tożsamości za pomocą czytnika linii papilarnych
- Dane dotyczące skanów są poufne

# TouchID

- Wykorzystywane do potwierdzania tożsamości za pomocą czytnika linii papilarnych
- Dane dotyczące skanów są poufne
- Dostępne na iPhone 5s, 6, 6+, iPad Air 2, iPad mini 3

# TouchID

- Wykorzystywane do potwierdzania tożsamości za pomocą czytnika linii papilarnych
- Dane dotyczące skanów są poufne
- Dostępne na iPhone 5s, 6, 6+, iPad Air 2, iPad mini 3
- Może nie być skonfigurowane na urządzeniu

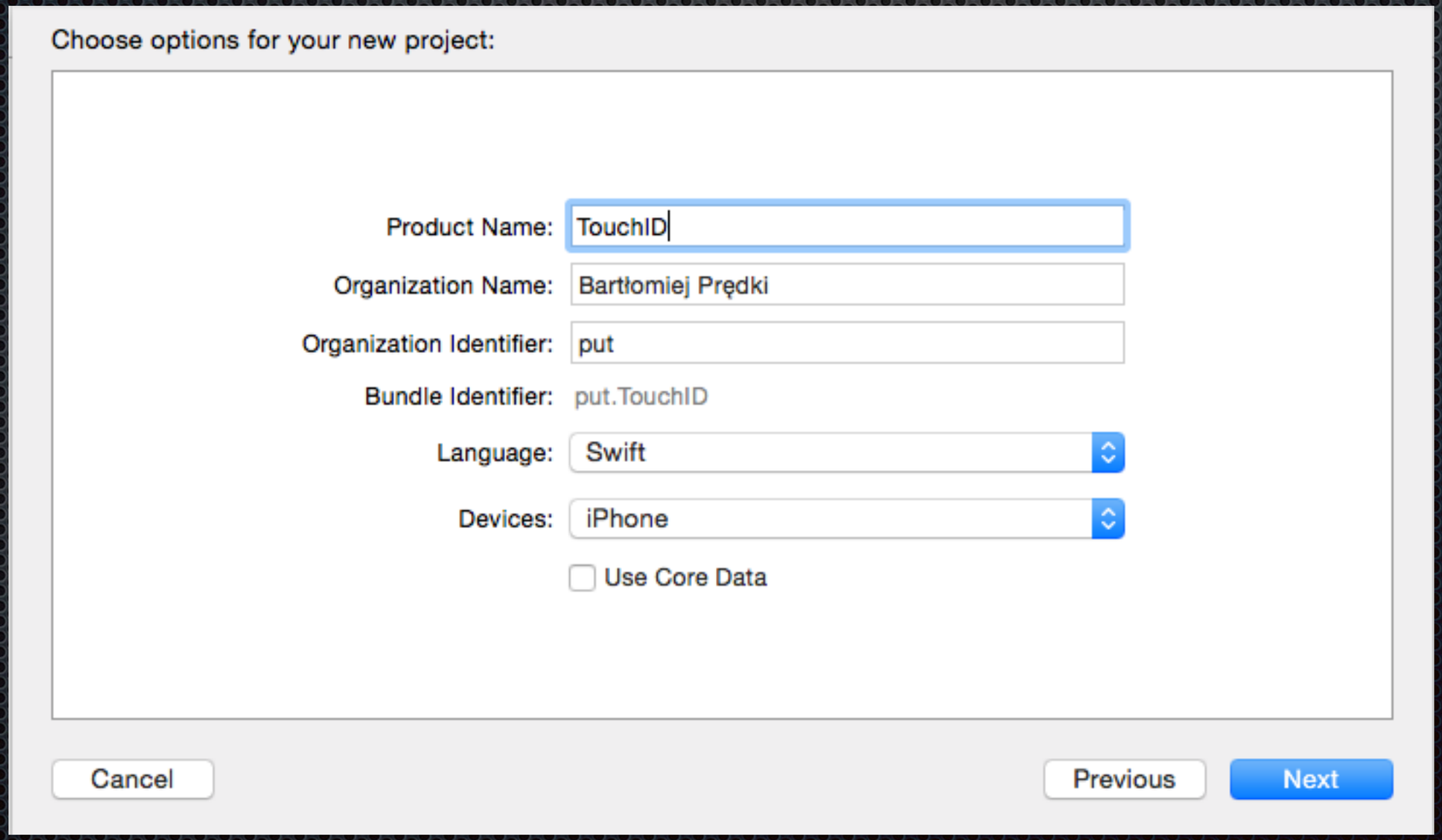
# TouchID

# TouchID

- Nowy projekt z szablonu Single View

# TouchID

- Nowy projekt z szablonu Single View



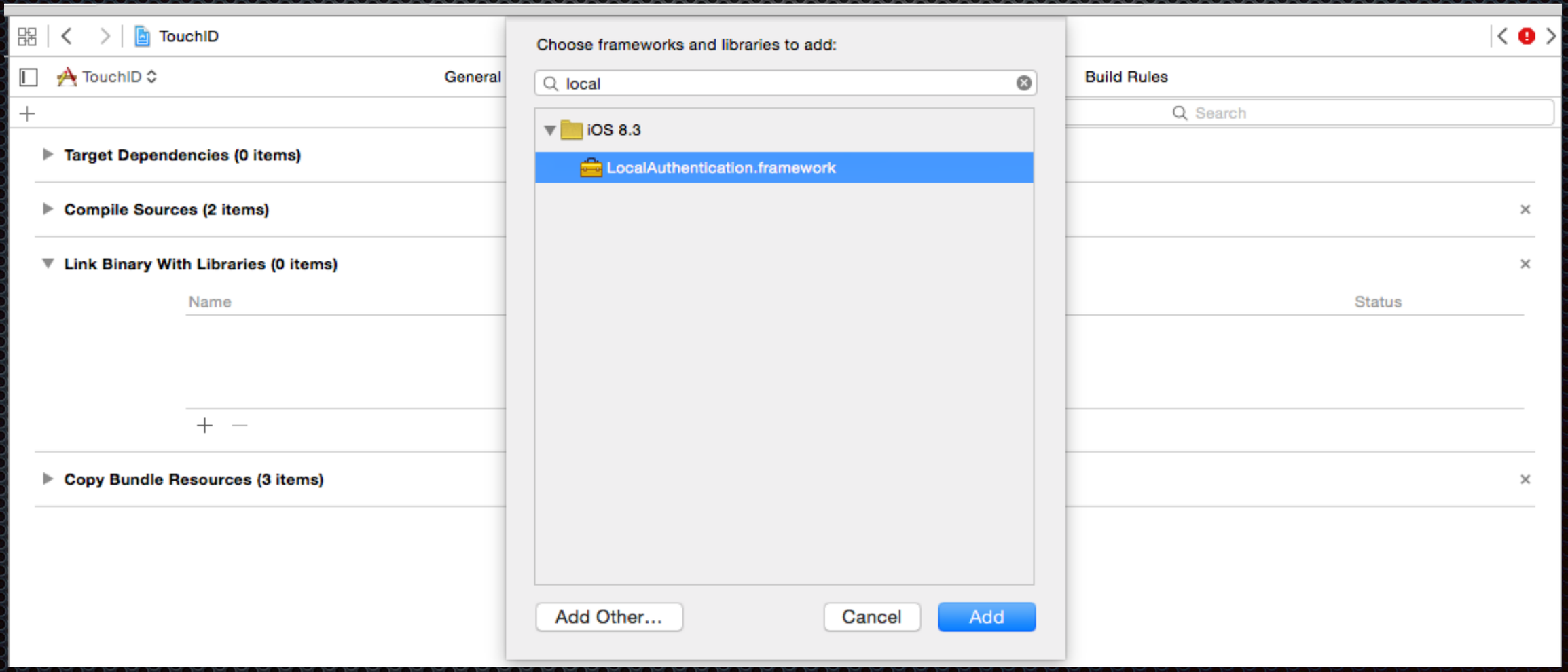
# TouchID

# TouchID

- Dodajemy bilbiotekę

# TouchID

- Dodajemy bilbiotekę



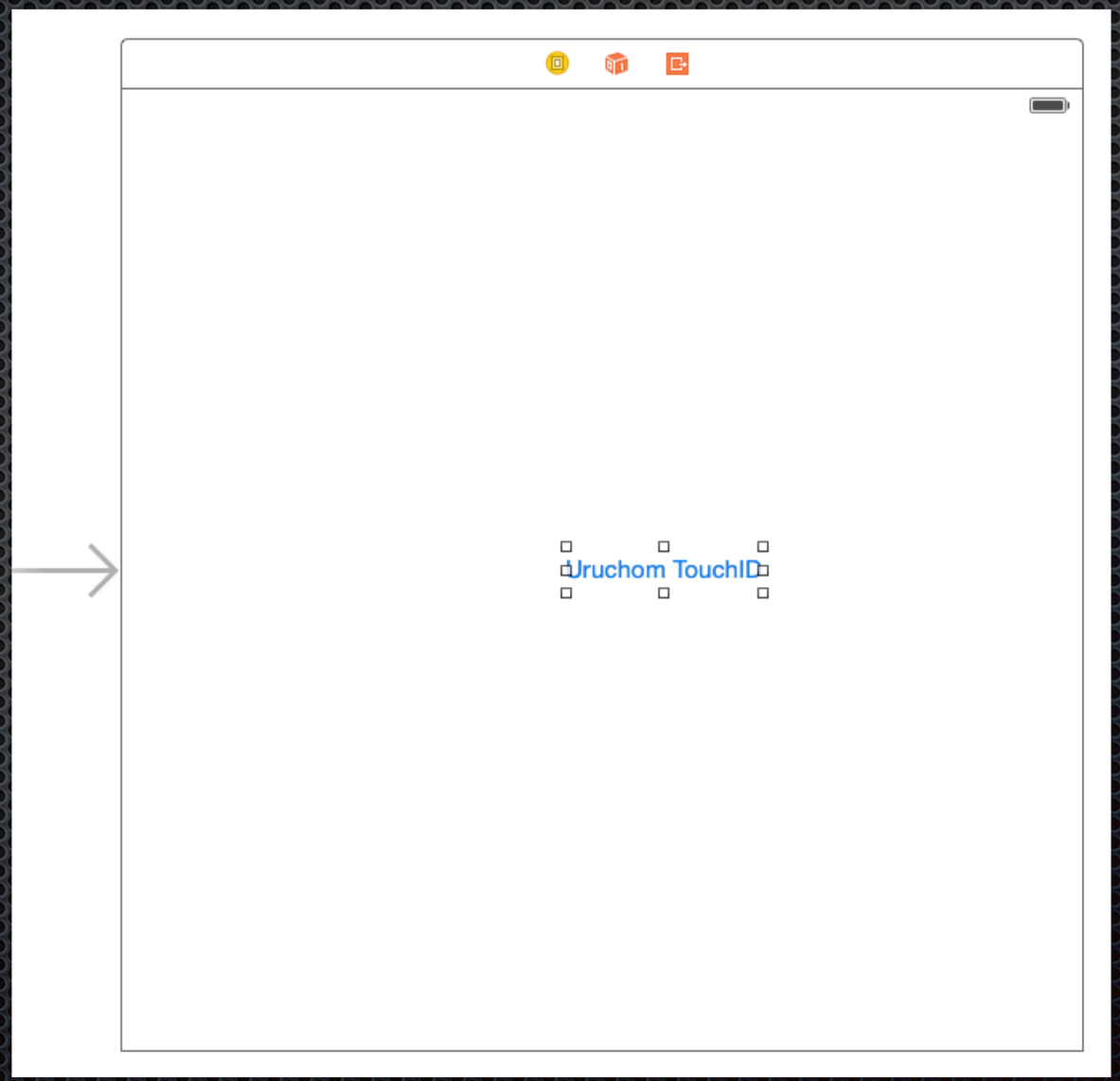
# TouchID

# TouchID

- W scenorysie dodajemy przycisk na środku ekranu

# TouchID

- W scenorysie dodajemy przycisk na środku ekranu



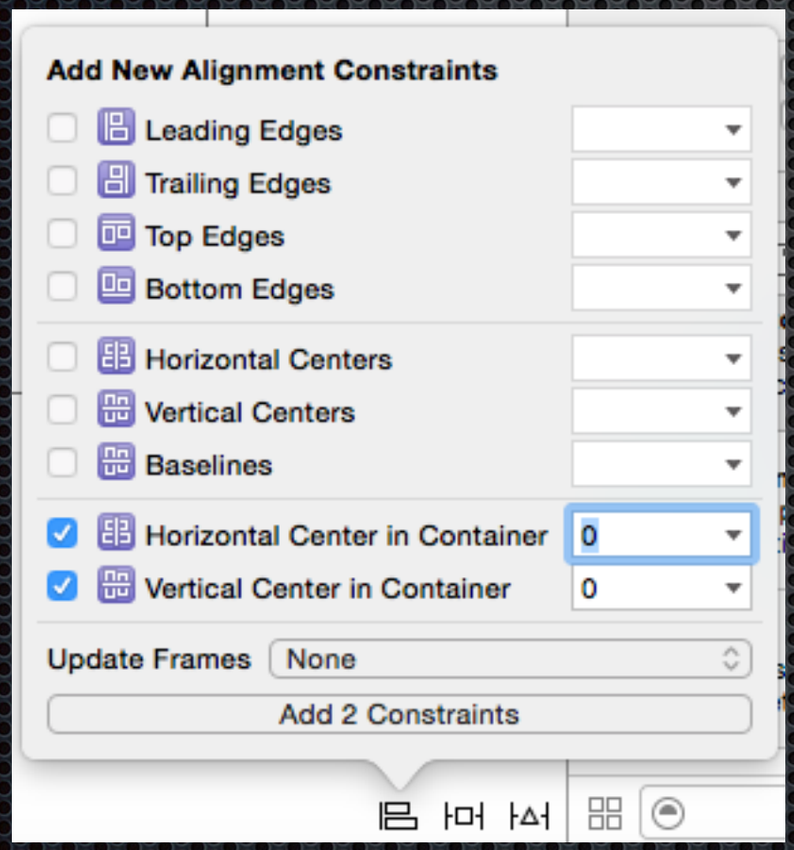
# TouchID

# TouchID

- Dodajemy wyrównanie do środka ekranu

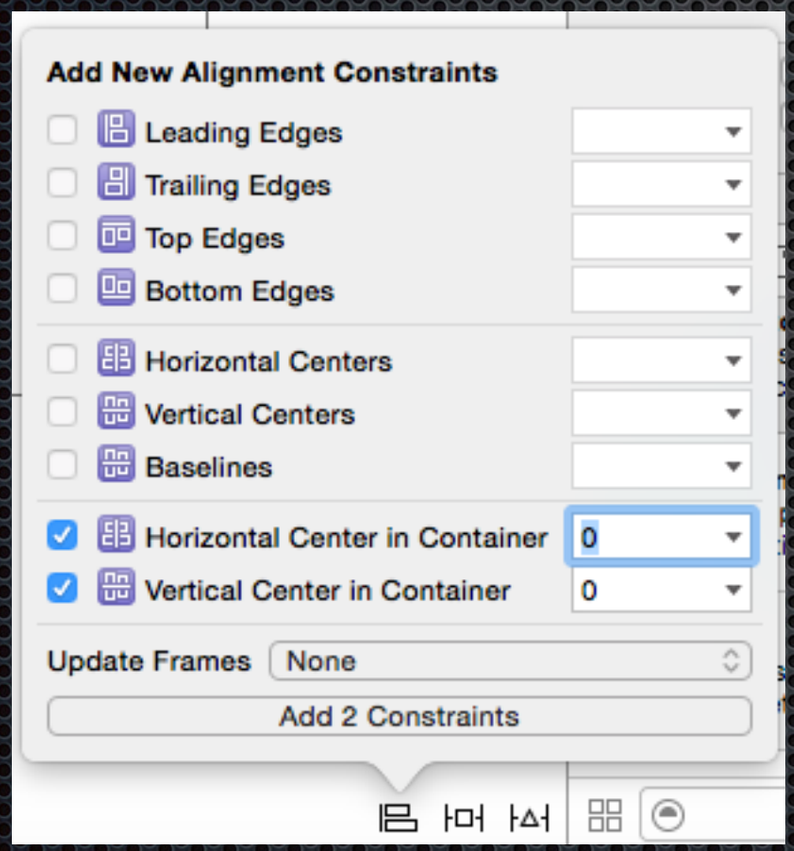
# TouchID

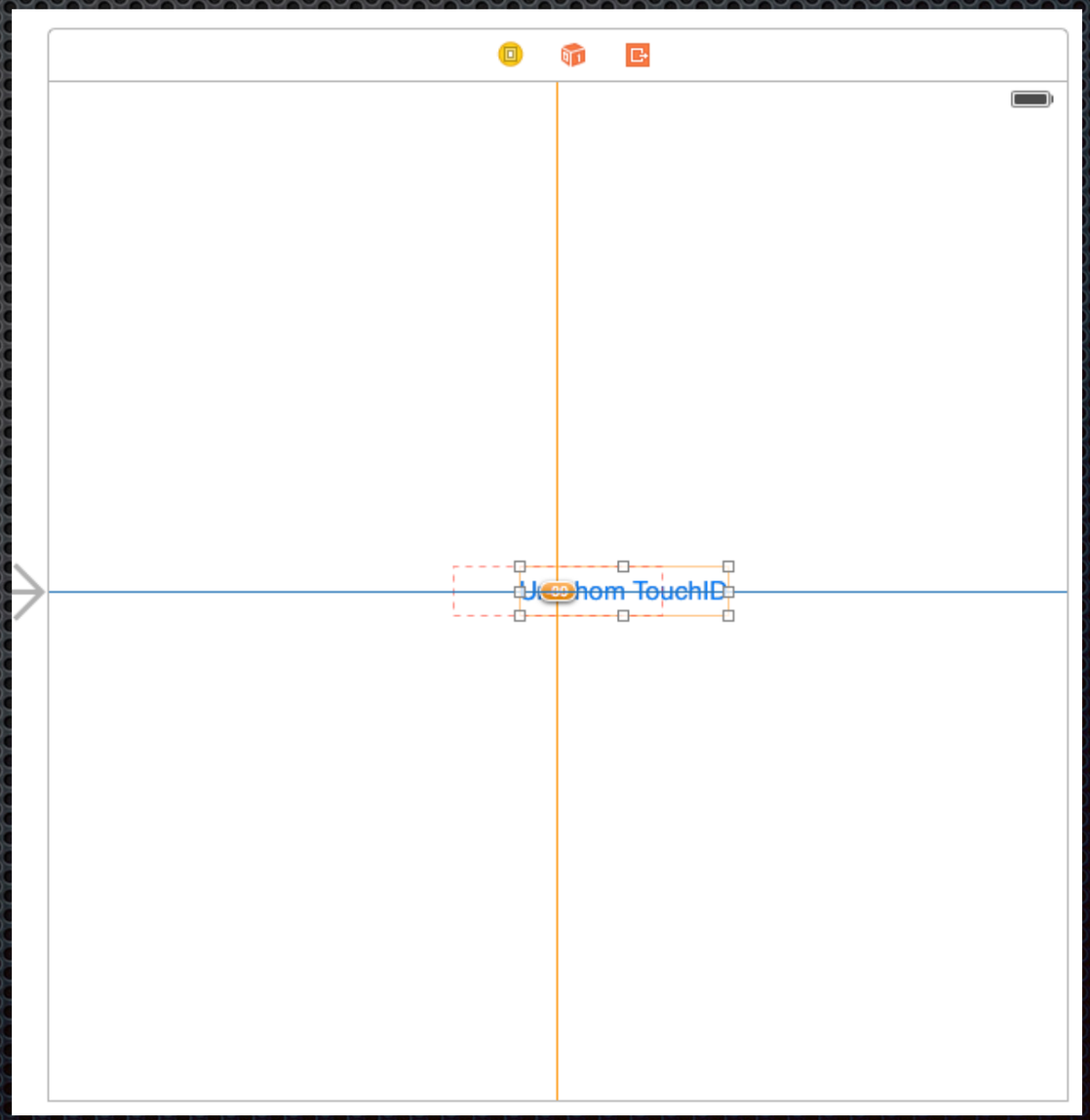
- Dodajemy wyrównanie do środka ekranu



# TouchID

- Dodajemy wyrównanie do środka ekranu



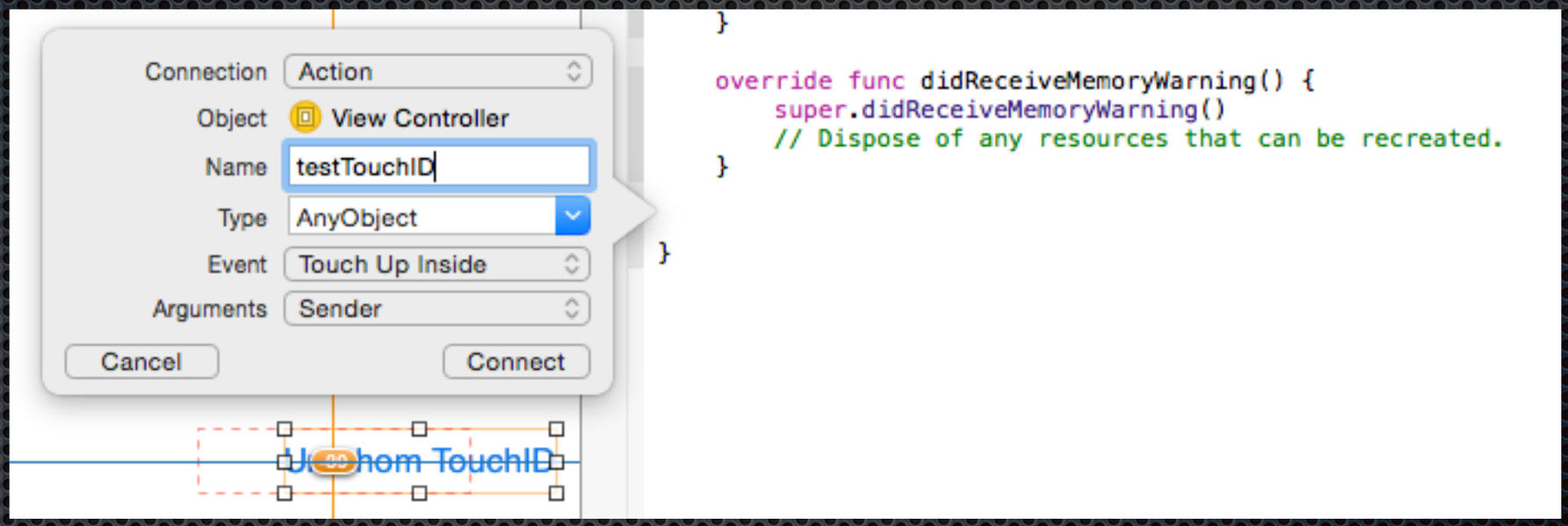
# TouchID

# TouchID

- Dodajemy akcję `testTouchID`

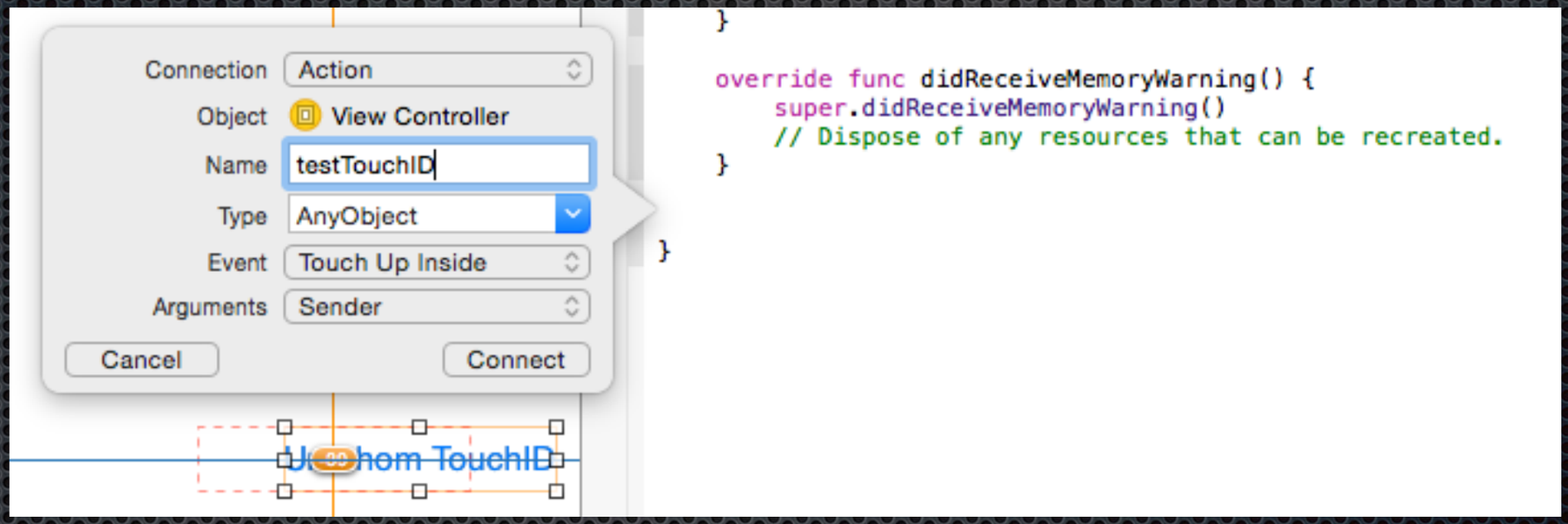
# TouchID

- Dodajemy akcję `testTouchID`



# TouchID

- Dodajemy akcję `testTouchID`




```
@IBAction func testTouchID(sender: AnyObject) {
}
```

# TouchID

# TouchID

- Dodajemy import biblioteki `LocalAuthentication`

```
import UIKit
import LocalAuthentication
```

# TouchID

- Dodajemy import biblioteki `LocalAuthentication`

```
import UIKit
import LocalAuthentication
```

- Dopisujemy metodę `notifyUser`

```
func notifyUser(msg: String, err: String?) {
    let alert = UIAlertController(title: msg,
        message: err,
        preferredStyle: UIAlertControllerStyle.Alert)

    let cancelAction = UIAlertAction(title: "OK",
        style: .Cancel, handler: nil)

    alert.addAction(cancelAction)

    self.presentViewController(alert, animated: true,
        completion: nil)
}
```

# TouchID

# TouchID

- Sprawdzamy, czy urządzenie wspiera TouchID

# TouchID

- Sprawdzamy, czy urządzenie wspiera TouchID

```
@IBAction func testTouchID(sender: AnyObject) {
    let context = LAContext()

    var error: NSError?

    if context.canEvaluatePolicy(
        LAPolicy.DeviceOwnerAuthenticationWithBiometrics,
        error: &error)
    {
            // Możemy użyć TouchID
    }

    else {
        // Nie możemy użyć TouchID
        switch error!.code{

        case LAError.TouchIDNotEnrolled.rawValue:
            notifyUser("TouchID nie jest włączone",
                err: error?.localizedDescription)

        case LAError.PasscodeNotSet.rawValue:
            notifyUser("Nie ustawiono hasła",
                err: error?.localizedDescription)

        default:
            notifyUser("TouchID niedostępne",
                err: error?.localizedDescription)

        }
    }
}
```

# TouchID

# TouchID

- Dopisujemy kod autoryzacji

# TouchID

- Dopisujemy kod autoryzacji

```
    // Możemy użyć TouchID
let myLocalizedReasonString = "Dostęp wymaga autoryzacji"
context.evaluatePolicy(
    LAPolicy.DeviceOwnerAuthenticationWithBiometrics,
    localizedReason: myLocalizedReasonString,
    reply:
    {
        (success, error) in

        if error != nil
        {

            switch error!.code
            {

            case LAError.SystemCancel.rawValue:
                self.notifyUser("Anulowano",
                    err: error?.localizedDescription)

            case LAError.UserCancel.rawValue:
                self.notifyUser("Spróbuj ponownie",
                    err: error?.localizedDescription)

            case LAError.UserFallback.rawValue:
                self.notifyUser("Autoryzacja",
                    err: "Wybrano hasło")

            default:
                self.notifyUser("Autoryzacja nieudana",
                    err: error?.localizedDescription)
            }

        }
        else
        {
            self.notifyUser("Autoryzacja powiodła się",
                err: "Masz pełen dostęp")
        }
})
```

# TouchID

# TouchID

- Emulator nie obsługuje TouchID

# TouchID

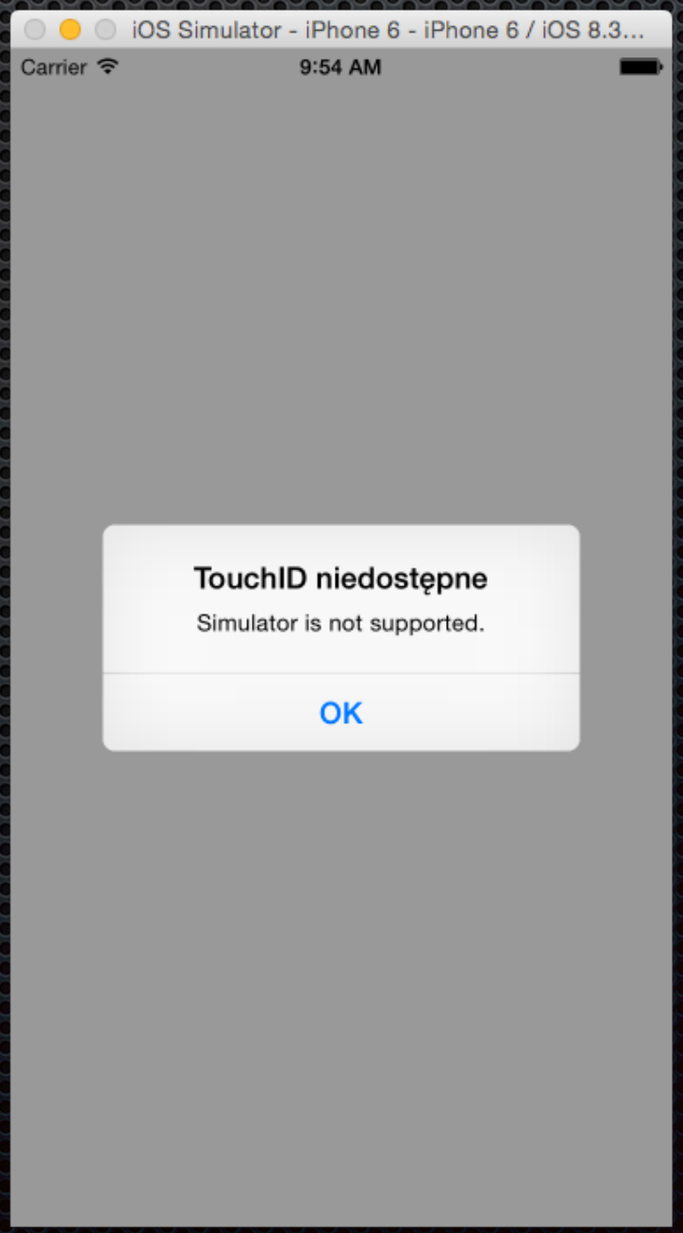
- Emulator nie obsługuje TouchID

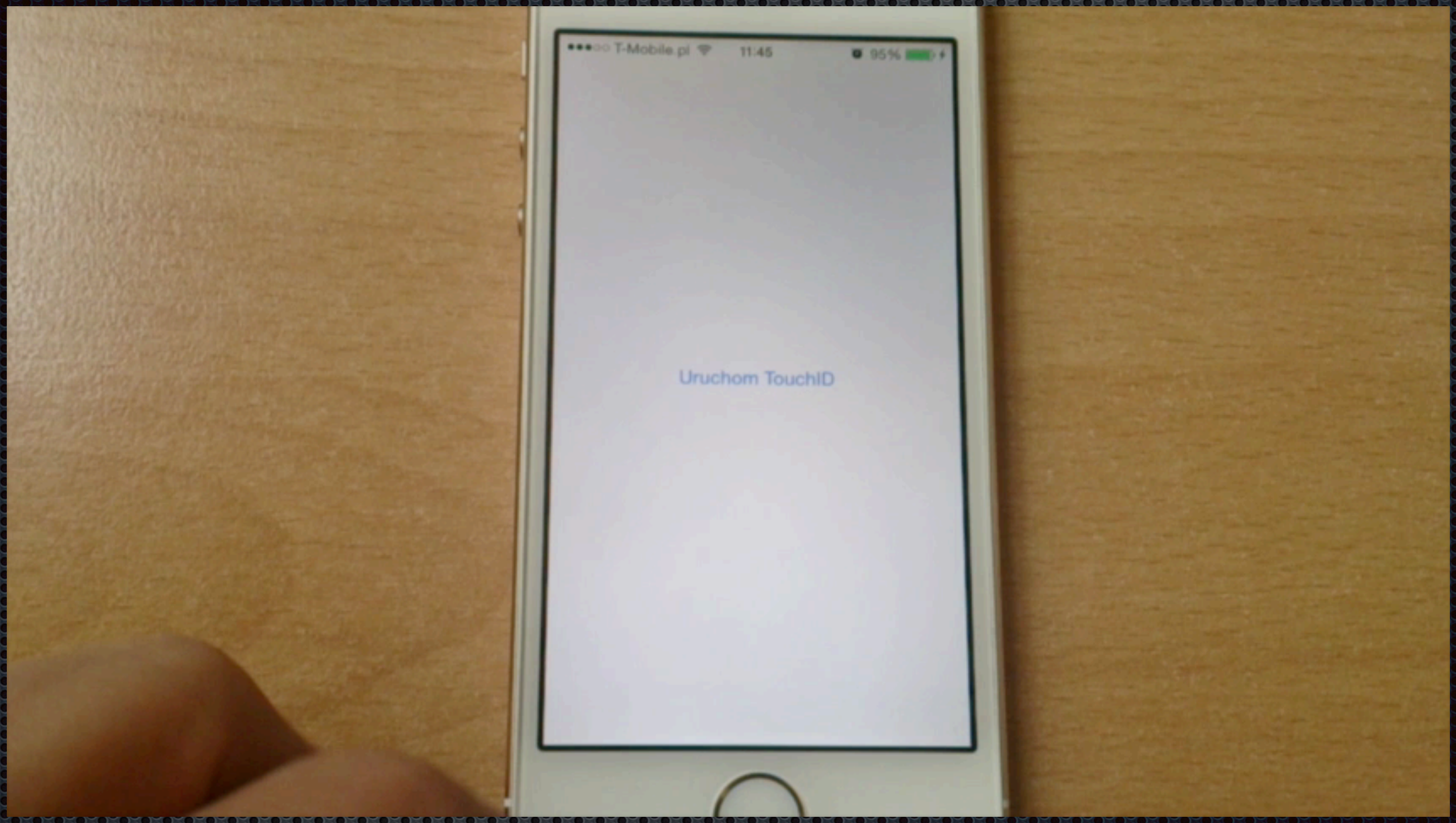


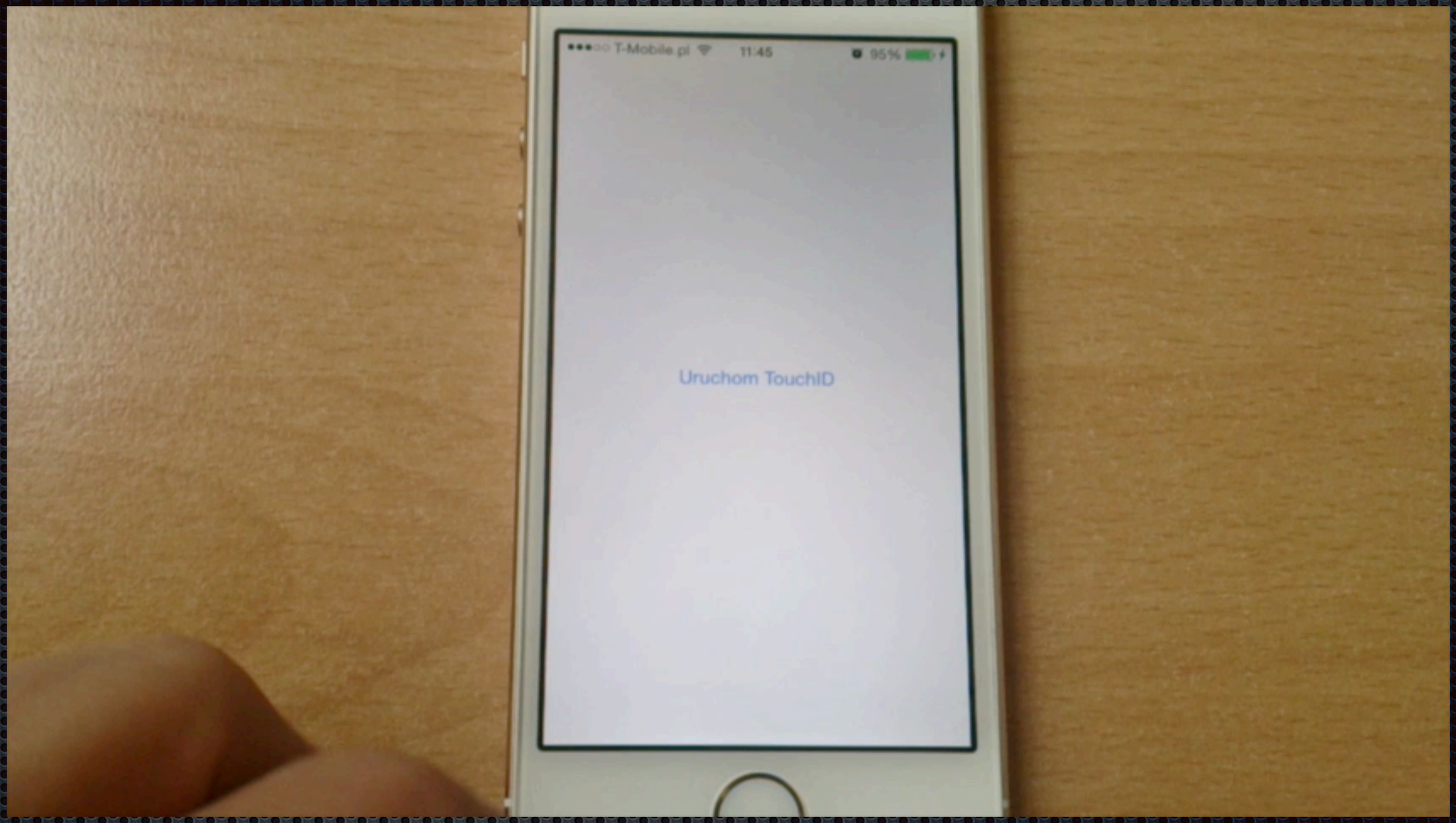
# TouchID

- Emulator nie obsługuje TouchID




T-Mobile.pl
11:45
95%
Uruchom TouchID


T-Mobile.pl
11:45
95%
Uruchom TouchID

# Do zobaczenia

速