



FAKULTET INŽENJERSKIH NAUKA UNIVERZITET U KRAGUJEVCU

Tema: Programiranje aplikacija iOS sistema
Student: Nevena Stašić
Profesor: Nenad Grujović

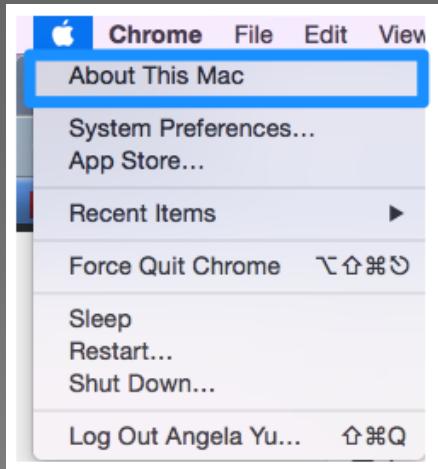
Kragujevac, 2019. godine

Instaliranje programa

- xCode je integrisano razvojno okruženje razvijeno od strane Apple kompanije, a ogromna većina iOS programera se oslanja na ovo okruženje za izradu iPhone ili iPad aplikacija. xCode 10 može da se instalira samo na MAC operativnom sistemu MacOS 10.13.4 (High Sierra) ili nekom novijem, ali je idealno da koristimo macOS 10.14.0 ili noviju (Mojave).
- Ukoliko još uvek imate neki stariji operativni sistem, moraćete da ažurirate taj operativni sistem.
- Da biste proverili trenutnu verziju operativnog sistema, idite na jabuku u gornjem levom uglu Mac, a potom na About This Mac. Ovaj proces je prikazan na slici u nastavku teksta.

Instaliranje programa

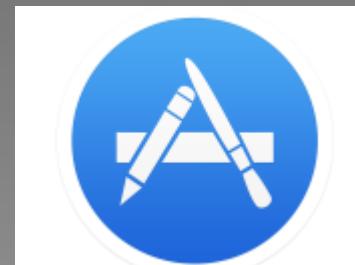
- Posle otvaranja polja About This Mac trebalo bi da vidite ekran sličan ovome:



Slike 1 i 2 – Provera verzije sistema

Instaliranje programa

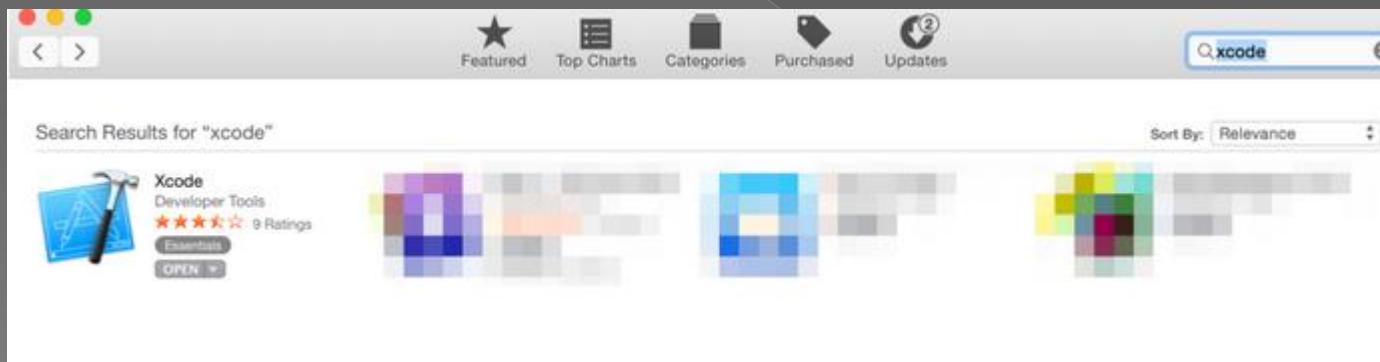
- Kada ste sigurni da koristite ispravnu verziju operativnog sistema Mac, možete početi sa preuzimanjem xCode 10 preko Mac App prodavnice. Otvorite aplikaciju App Store na Mac računaru. Po podrazumevanoj vrednosti App Store je u Dock-u, liniji na dnu ekrana. Možete ga pronaći i na vašoj lansiranoj podlozi.



Slika 3 – App Store

Instaliranje programa

- U polju za pretragu u gornjem desnom uglu ukucajte xCode i pritisnite taster Return.



Slika 4 – Pretraga programa

Instaliranje programa

- xCode je besplatna aplikacija koju je razvila kompanija Apple, pa samo kliknite na dugme Get ili Download i pokrenite proces instalacije.
- xCode je veličine nekoliko gigabajta tako da preuzimanje može potrajati. Podrazumevano se xCode preuzima u direktorijum Applications.



Slika 5 – Pokretanje preuzimanja aplikacije

Instaliranje programa

- Da bismo mogli da započnemo proces preuzimanja moramo imati apple ID, kojim ćemo biti prijavljeni na App Store. U slučaju da nemamo apple ID, veoma lako i brzo ga možemo kreirati na stranici:

<https://appleid.apple.com/account#!&page=create>

Izgled programa

xCode je programsko okruženje koje se koristi za programiranje iOS aplikacija.

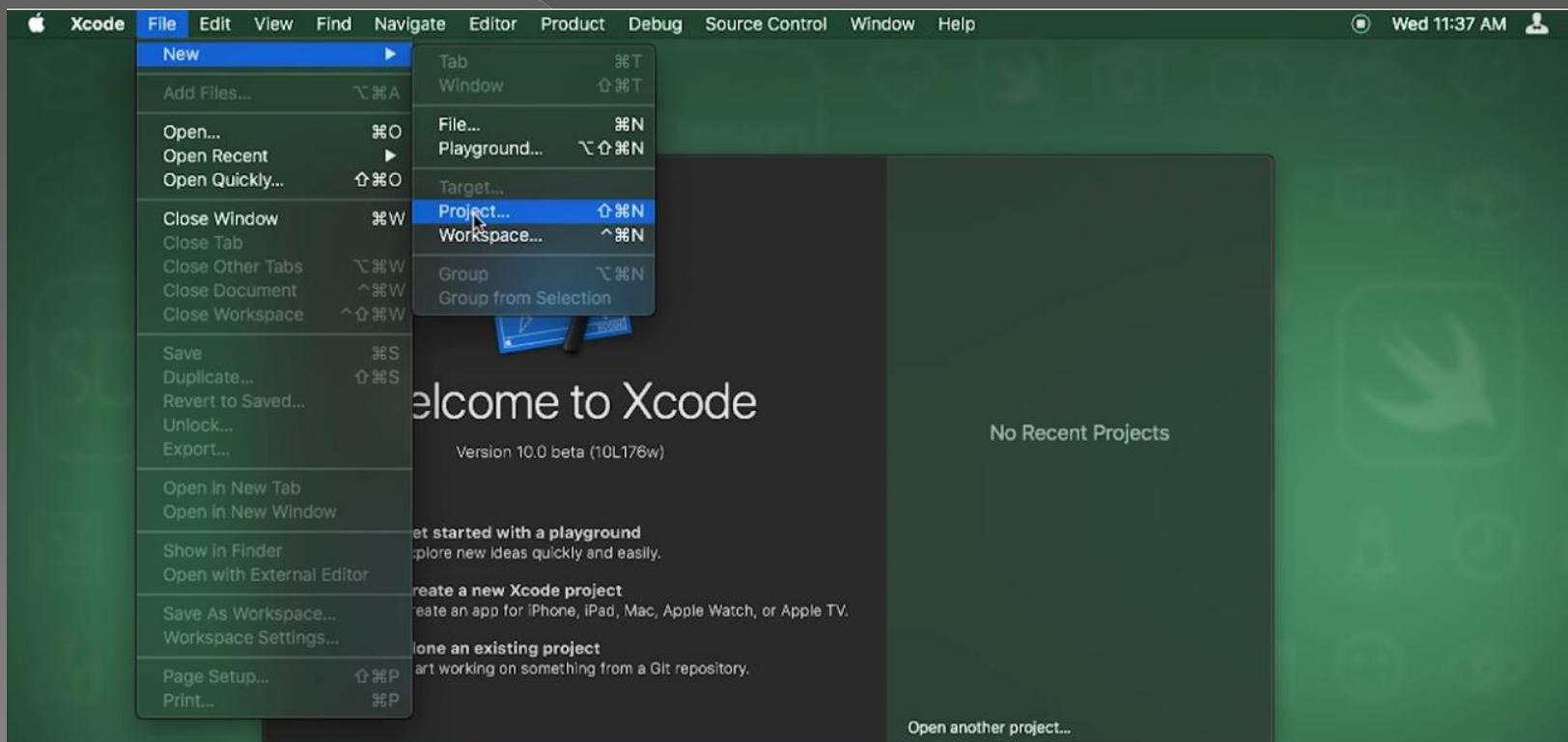
Izgled ovog programa je prikazan na slici 6, ispod.



Slika 6 – Izgled programa xCode

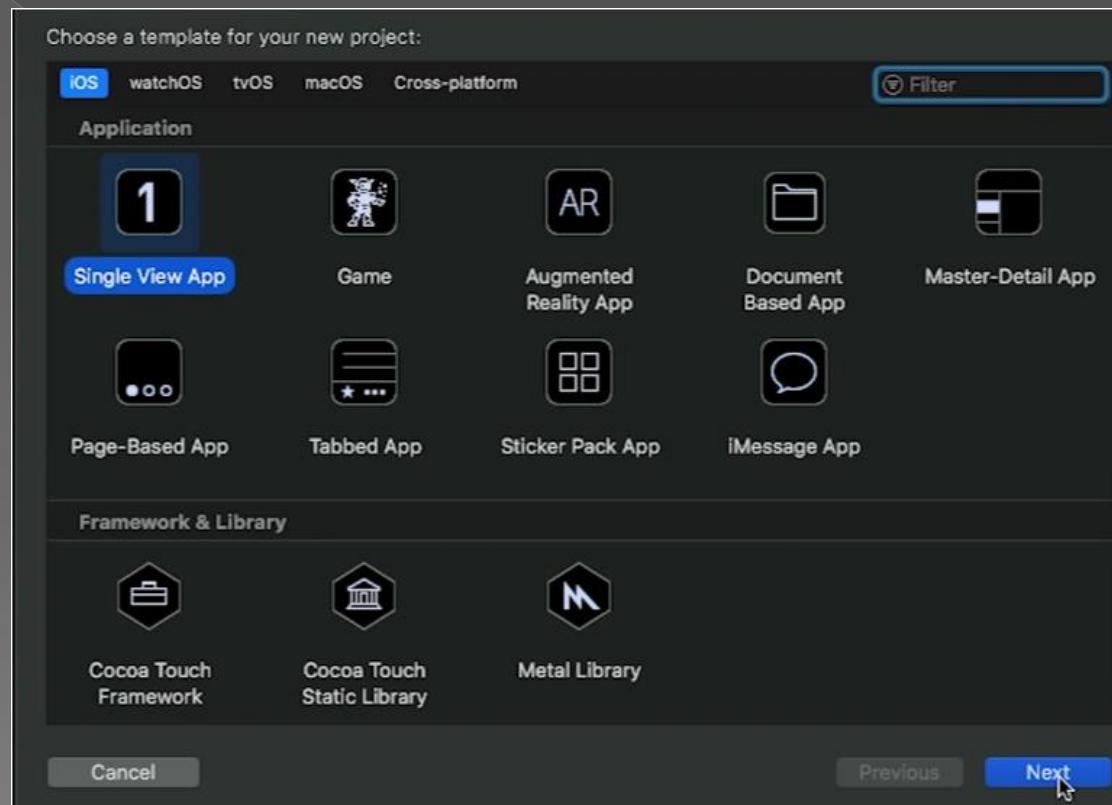
Kreiranje projekta

- Na slikama u nastavku, biće redno prikazan način kreiranja projekta u ovom programu.



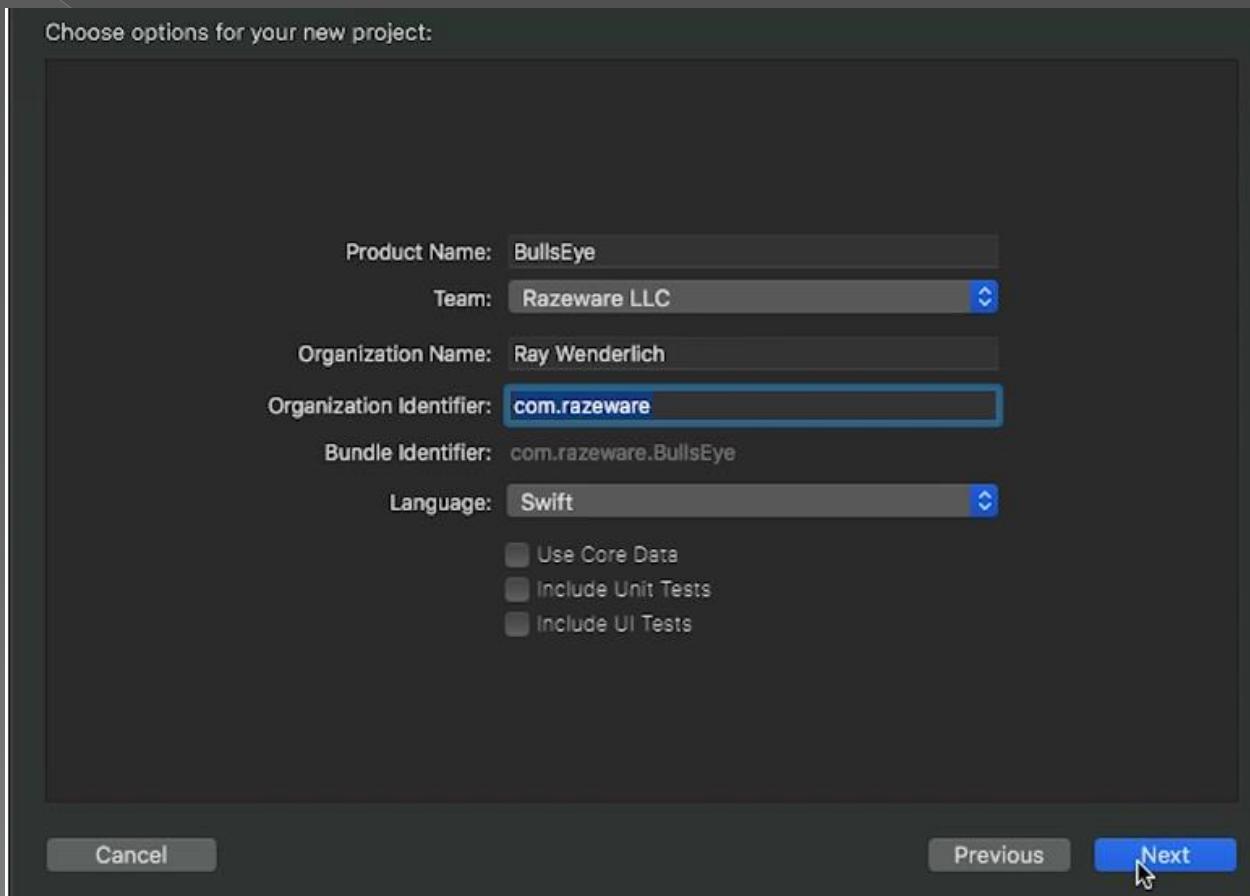
Slika 7 – Kreiranje novog projekta

Kreiranje projekta



Slika 8 – Kreiranje novog projekta

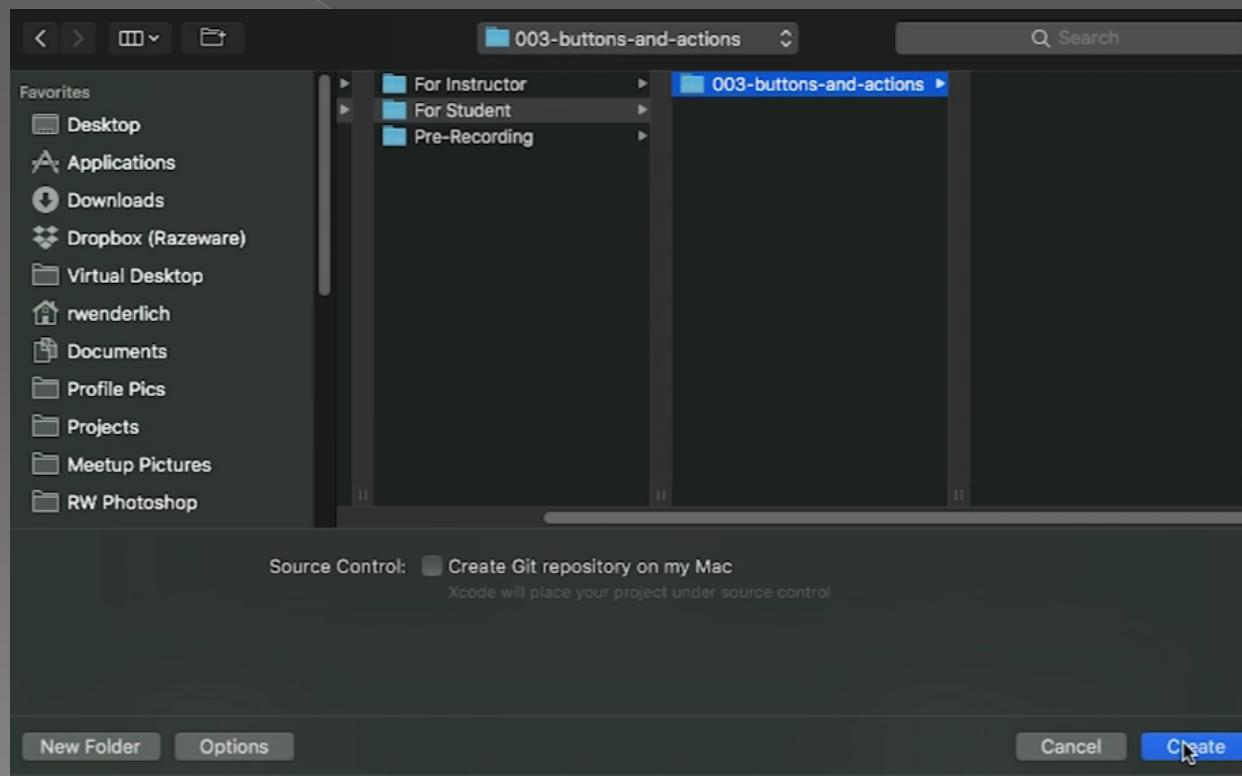
Kreiranje projekta



Slika 9 – Kreiranje novog projekta

Kreiranje projekta

- Biranje mesta na kome će projekat biti sačuvan.



Slika 10 – Kreiranje novog projekta

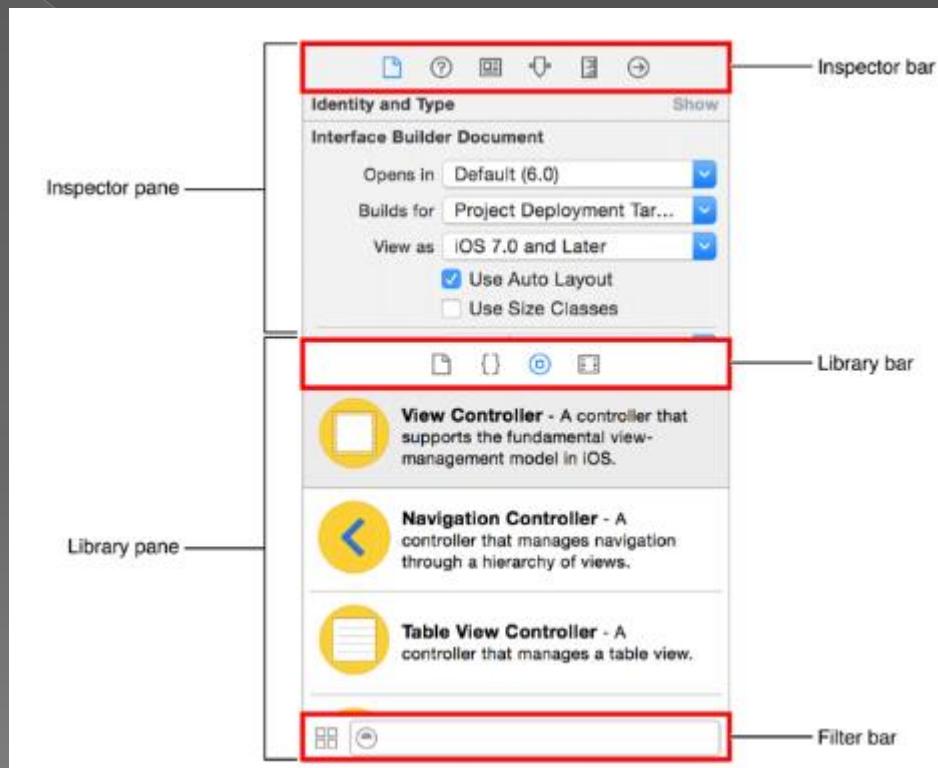
Izgled korisničkog dela programa

- Na slici ispod je prikazan izgled korisničkog dela programa. Sastoje se od četiri glavne komponente koje možemo prikazivati ili skrivati.
- Prvo na šta nailazimo, gledano od vrha ka dnu programa je Main Window Toolbar, koji je poprilično jednostavan. Unutar Toolbar-a možemo pokretati ili zaustavljati aplikaciju klikom na Run gumb, menjati uređaj na kojem ćemo simulirati rad aplikacije i menjati izgled samog korisničkog dela programa xCode-a.
- Ispod Main Window Toolbar-a, nalazi se Navigator Area. Unutar Navigator Area možemo pronaći sve datoteke koje se nalaze unutar našeg projekta. Unutar Navigator-a imamo 8 različitih pogleda koje prikazujemo odabirom jedne od ikona u tabovima na vrhu Navigatora, a to su redom: Project Navigator, Symbol Navigator, Find Navigator, Issue Navigator, Test Navigator, Debug Navigator, Breakpoint Navigator, Report Navigator.

Izgled korisničkog dela programa

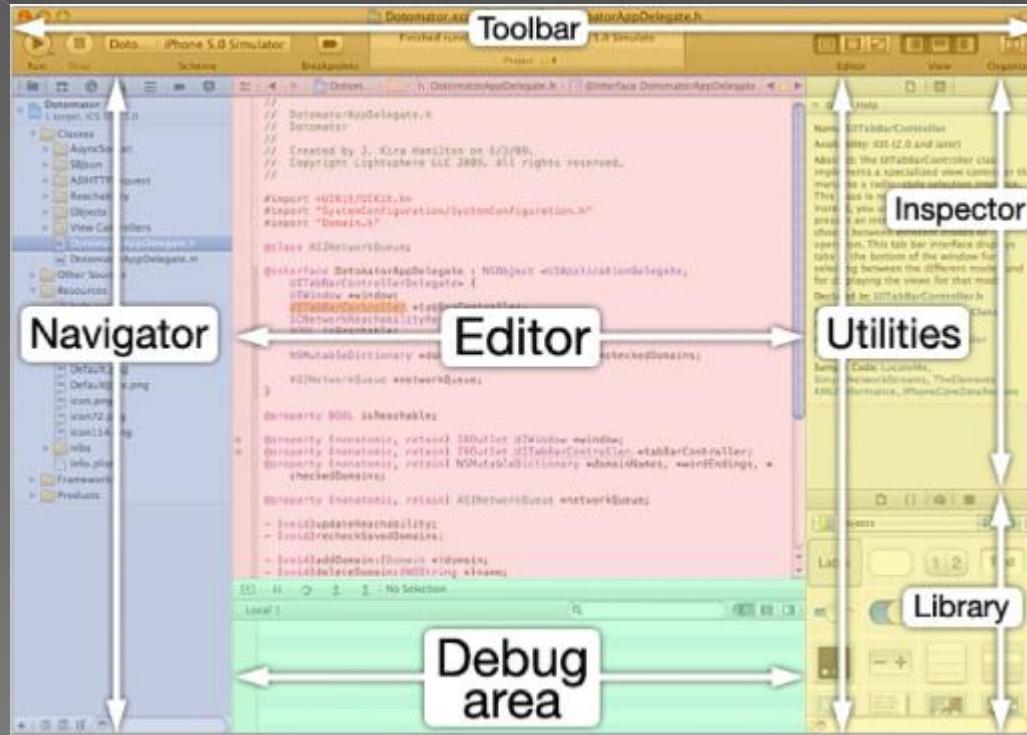
- Dalje na šta nailazimo posmatrajući programsko okruženje je Editor area, koji prikazuje sadržaj datoteke koja je odabrana u Navigatoru. Unutar glavnog dela se nalazi i editor unutar kojeg uređujemo sadržaj odabранe datoteke.
- Poslednje što se nalazi posmatrano sa vrha programa je Utilities Area, koji se sastoji od dva dela: Inspectore pane i Library pane. Unutar Inspector pane-a možemo pronaći inspectore, a unutar Library pane-a možemo pronaći resource library koje možemo koristiti unutar svog projekta. Objekte iz biblioteke koji su nam potrebni kreiramo drug and dropom na Editor Area.

Izgled korisničkog dela programa



Slika 11 – Deo programskog okruženja

Izgled korisničkog dela programa



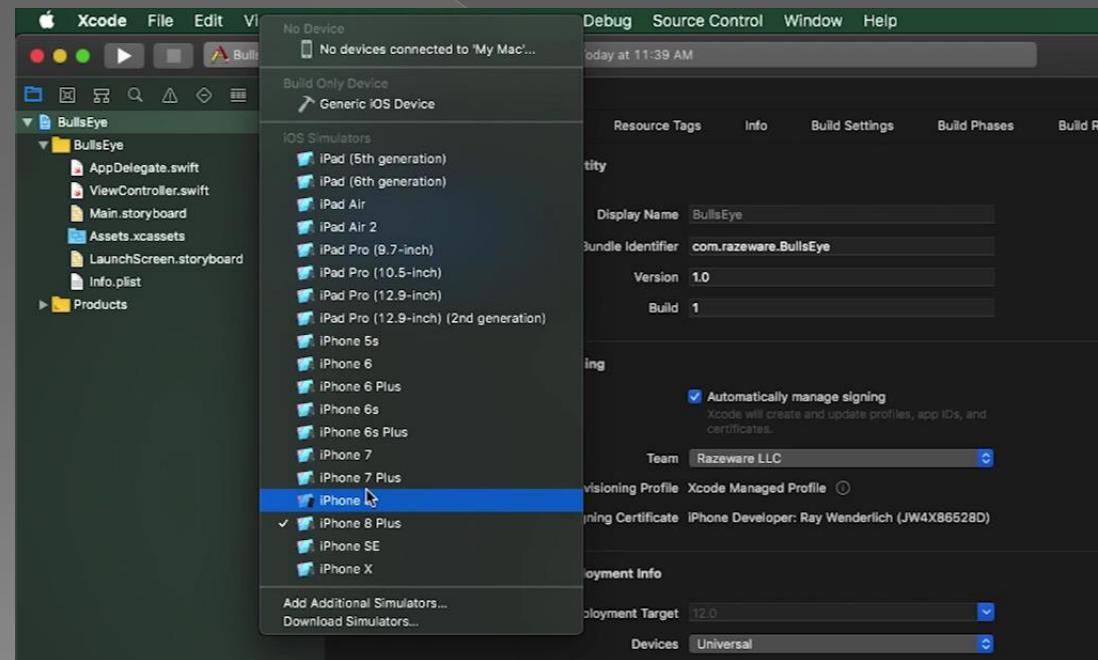
Slika 12 – Izgled programa po delovima

Podešavanje projekta

- Podešavanje veličine ekrana, određivanjem uređaja za koji se aplikacija pravi.



Slika 6 – Podešavanje projekta



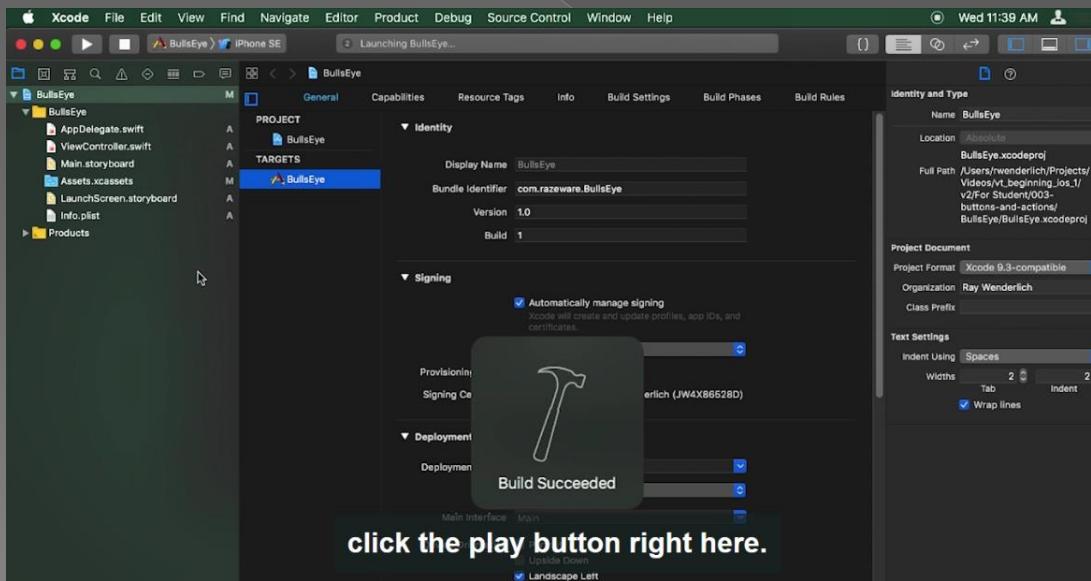
Slika 13 – Odabir uređaja za koji se pravi aplikacija 17

Pokretanje programa

- Kompajliranje i pokretanje programa se vrši na sledeći način:



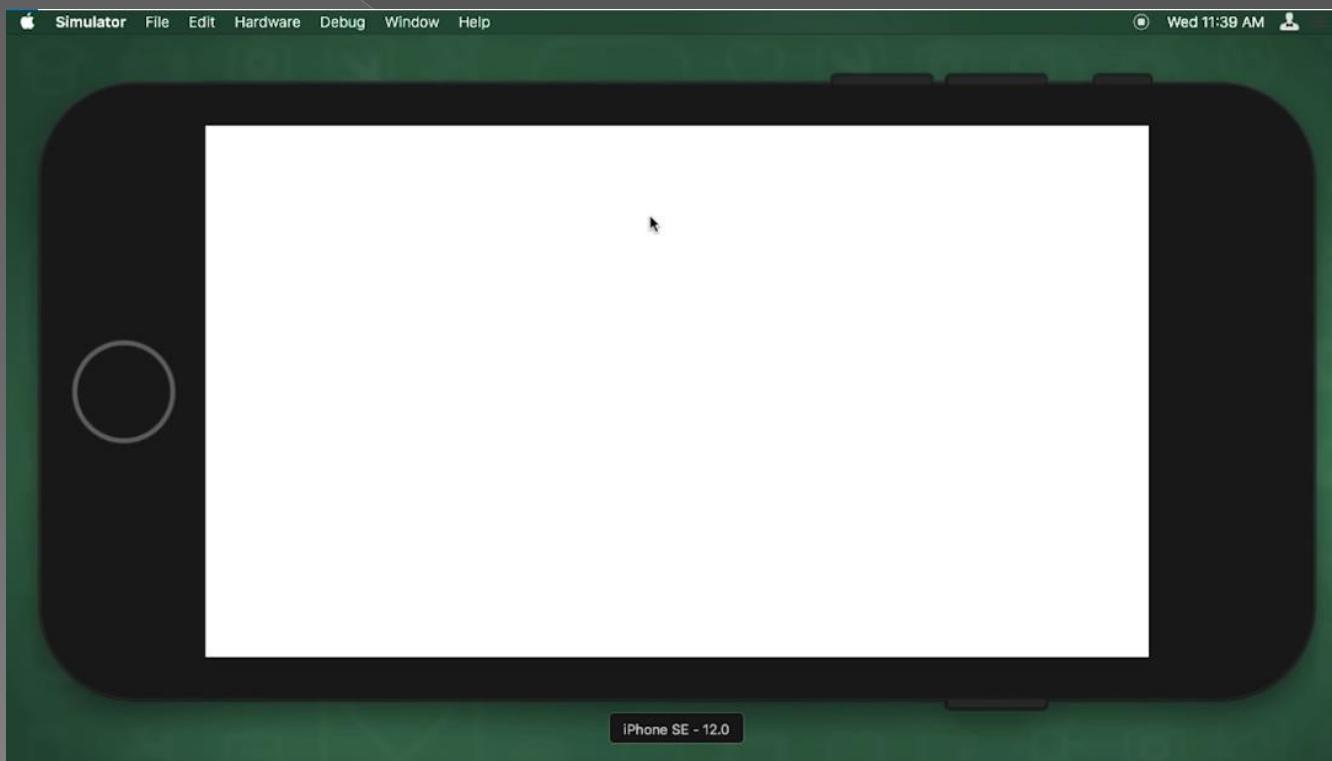
Slika 8 – Kompajliranje programa



Slika 14 – Uspešno kompajliranje programa

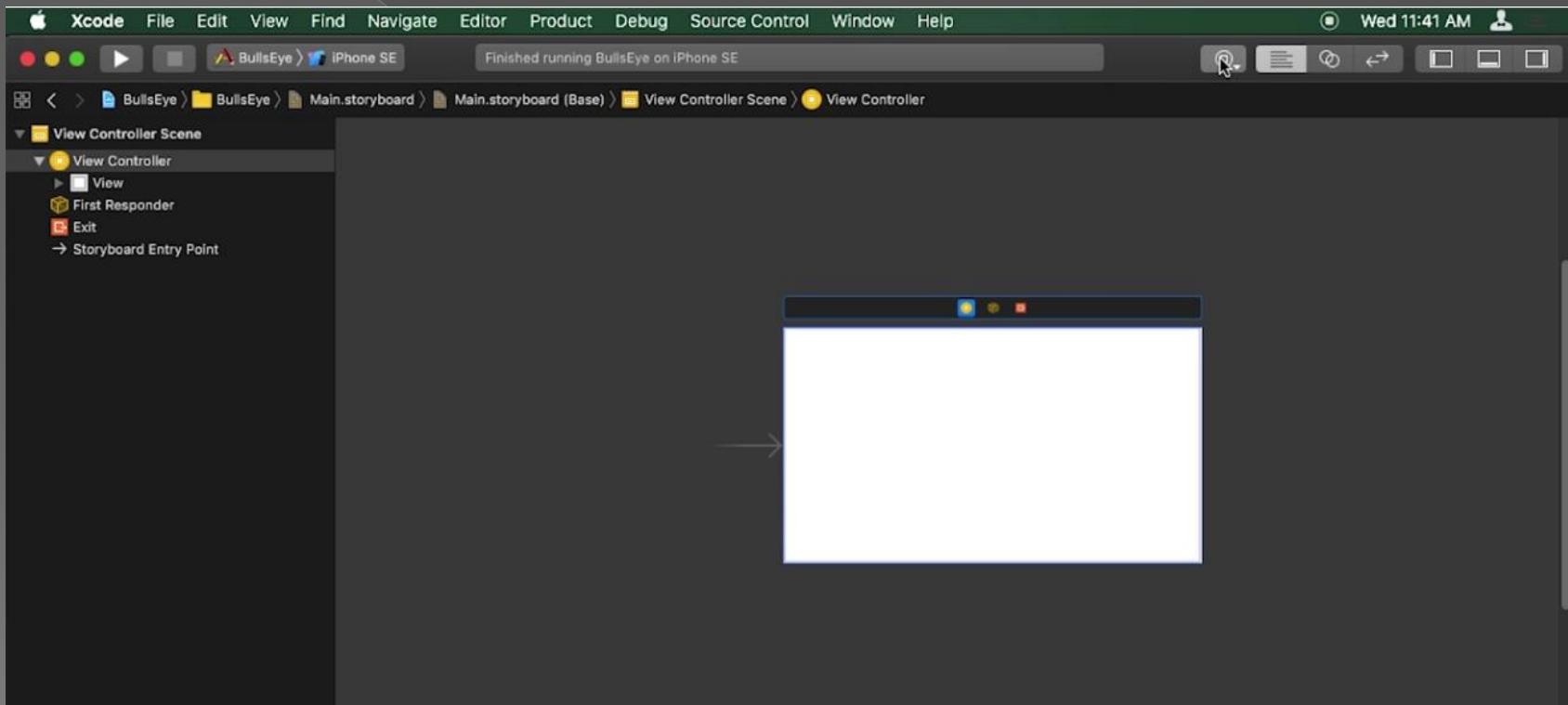
Pokretanje programa

- Izgled pokrenutog programa:



Slika 15 – Pokrenuti program

Izgled designer-a u kome se vrši razvoj programa



Slika 16 – Designer za razvoj aplikacije

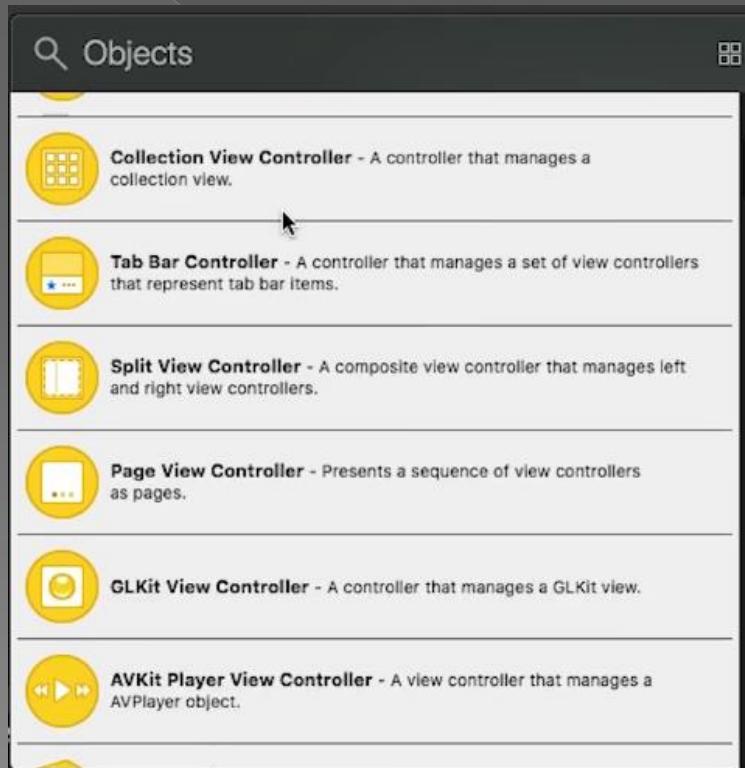
Dodavanje elemenata na prozor aplikacije

- Klikom na ikonicu sa slike ispod, otvara se lista ponuđenih elemenata za dodavanje na ekran aplikacije. Dodavanje nekog od tih elementa vrši se prevlačenjem odabranog elementa na prozor aplikacije.
- Radi lakše pretrage konkretnog elementa koji nam je potreban, postoji polje pretrage u koje možemo uneti naziv elementa koji nam treba.
- Sve ove mogućnosti biće konkretno pojašnjene i objašnjene uz pomoć slika, u nastavku prezentacije.

Dodavanje elemenata na prozor aplikacije

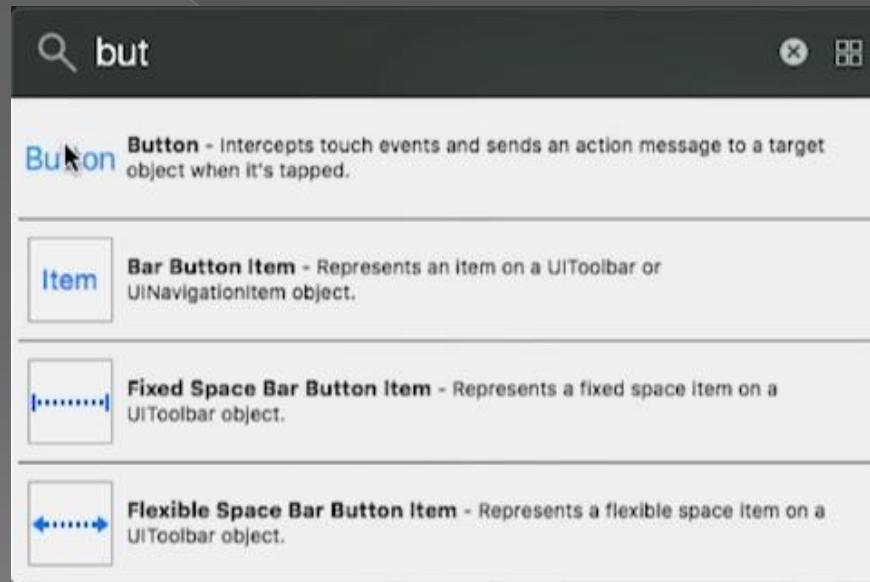


Dodavanje elemenata



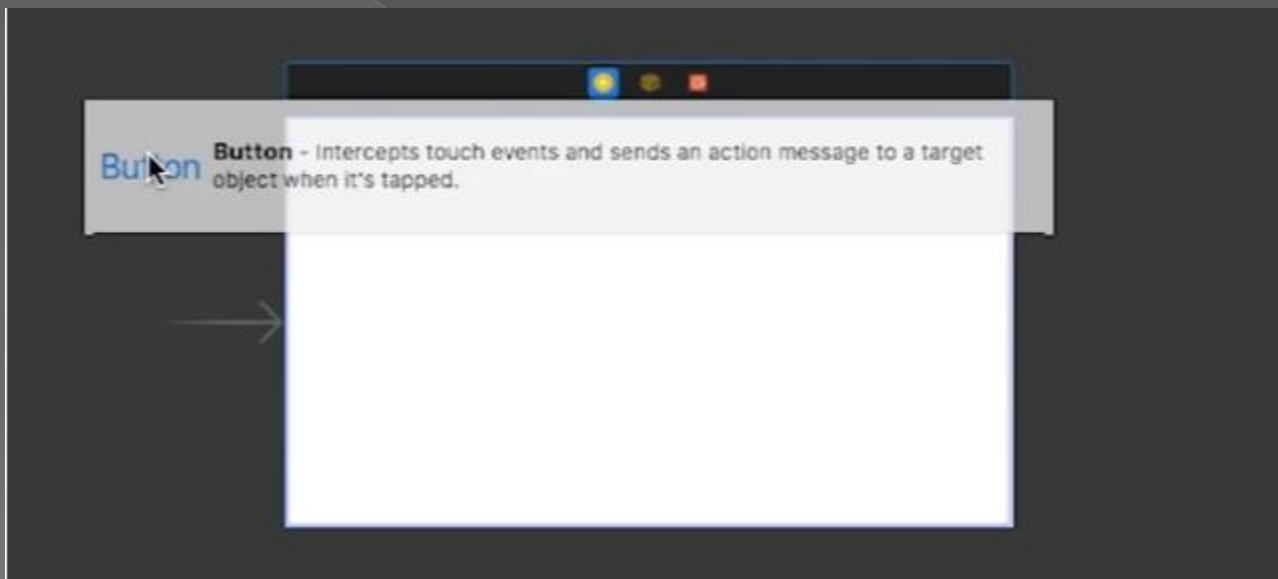
Slika 17 – Ponuđeni elementi

Dodavanje elemenata na prozor aplikacije



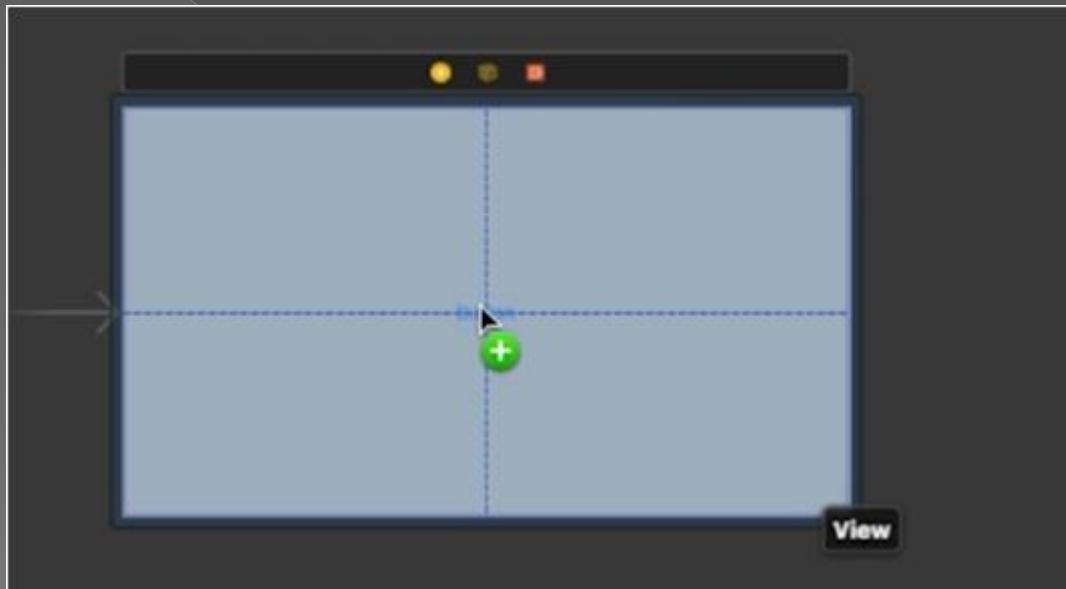
Slika 18 – Pretraga dostupnih elemenata

Dodavanje elemenata na prozor aplikacije



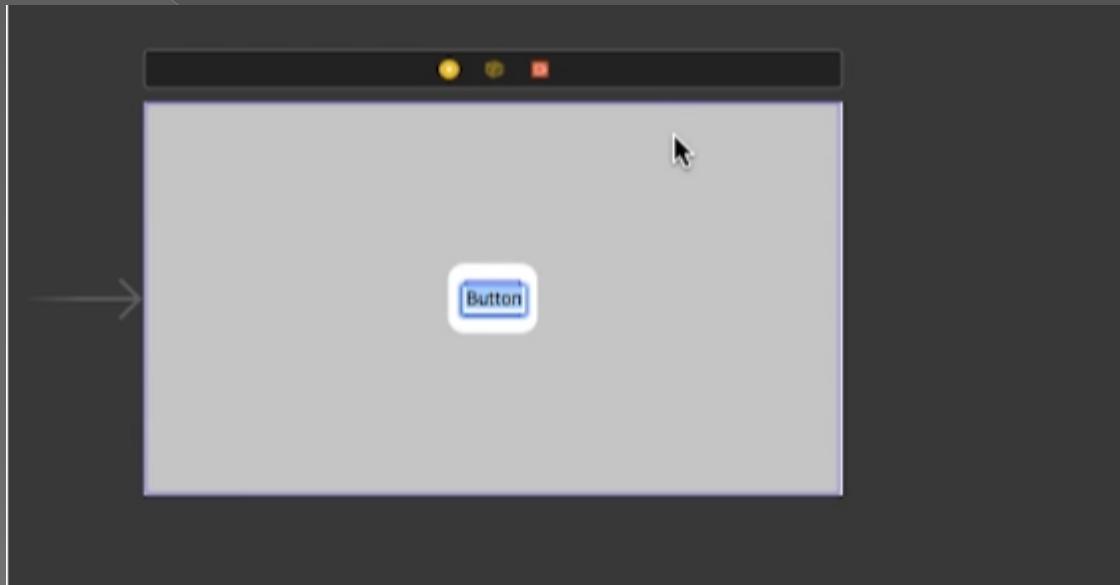
Slika 19 – Prevlačenje elementa u prozor aplikacije

Dodavanje elemenata na prozor aplikacije



Slika 20 – Dodavanje elementa na ekran

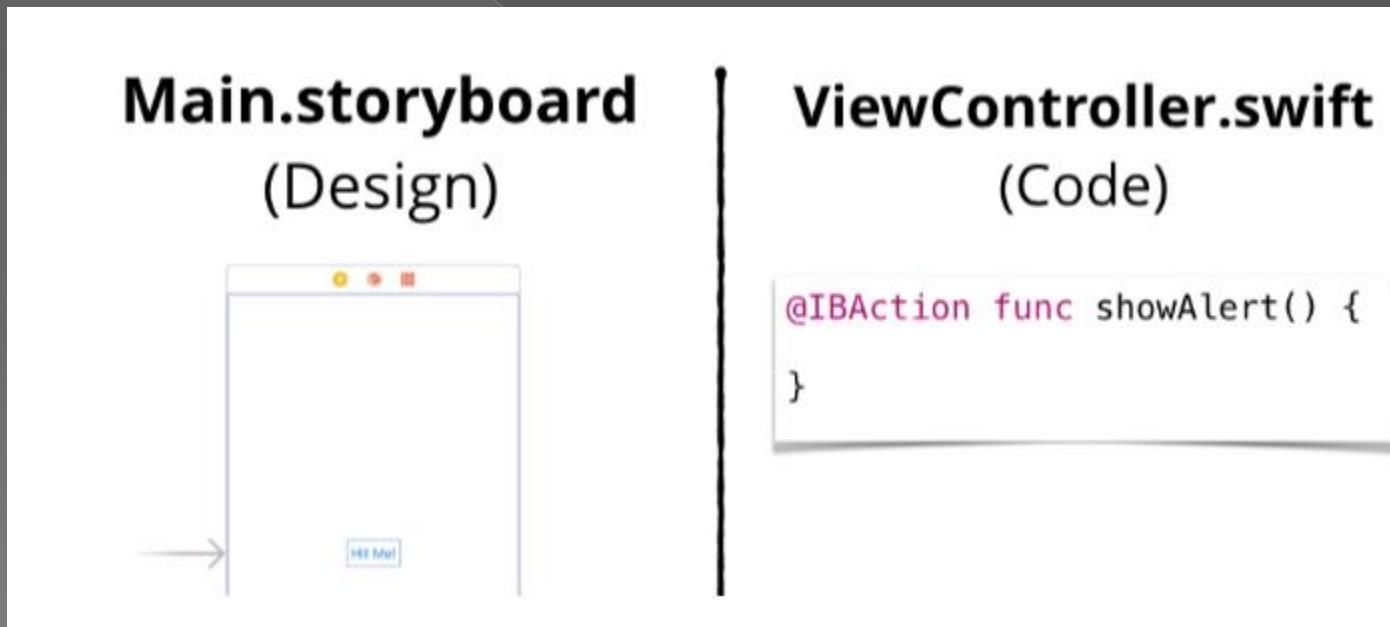
Dodavanje button-a



Slika 21 – Dodavanje button- a u prozor aplikacije

xCode programsko okruženje

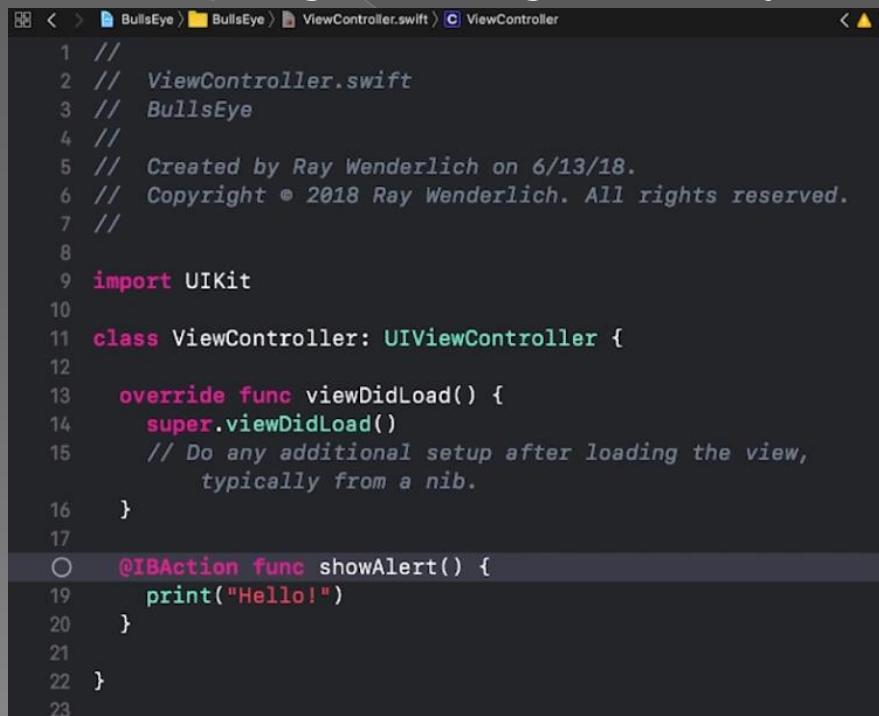
- Pregled fajlova u kome se nalazi design aplikacije i fajla u kome se nalazi programski kod aplikacije.



Slika 22 – Pravila xCode programskog okruženja

Povezivanje button-a sa nekom akcijom

- Primer koji sledi, predstavlja primer, u kome smo button koji smo postavili u prozor aplikacije povezali sa akcijom štampanja. Naime, klikom na button, štampaće nam se neka poruka u konzoli programskog okruženja.



The screenshot shows the Xcode interface with the file `ViewController.swift` open. The code defines a `ViewController` class that overrides `viewDidLoad()` and contains an `@IBAction` function `showAlert()` which prints "Hello!" to the console.

```
// ViewController.swift
// BullsEye
//
// Created by Ray Wenderlich on 6/13/18.
// Copyright © 2018 Ray Wenderlich. All rights reserved.

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view,
        // typically from a nib.
    }

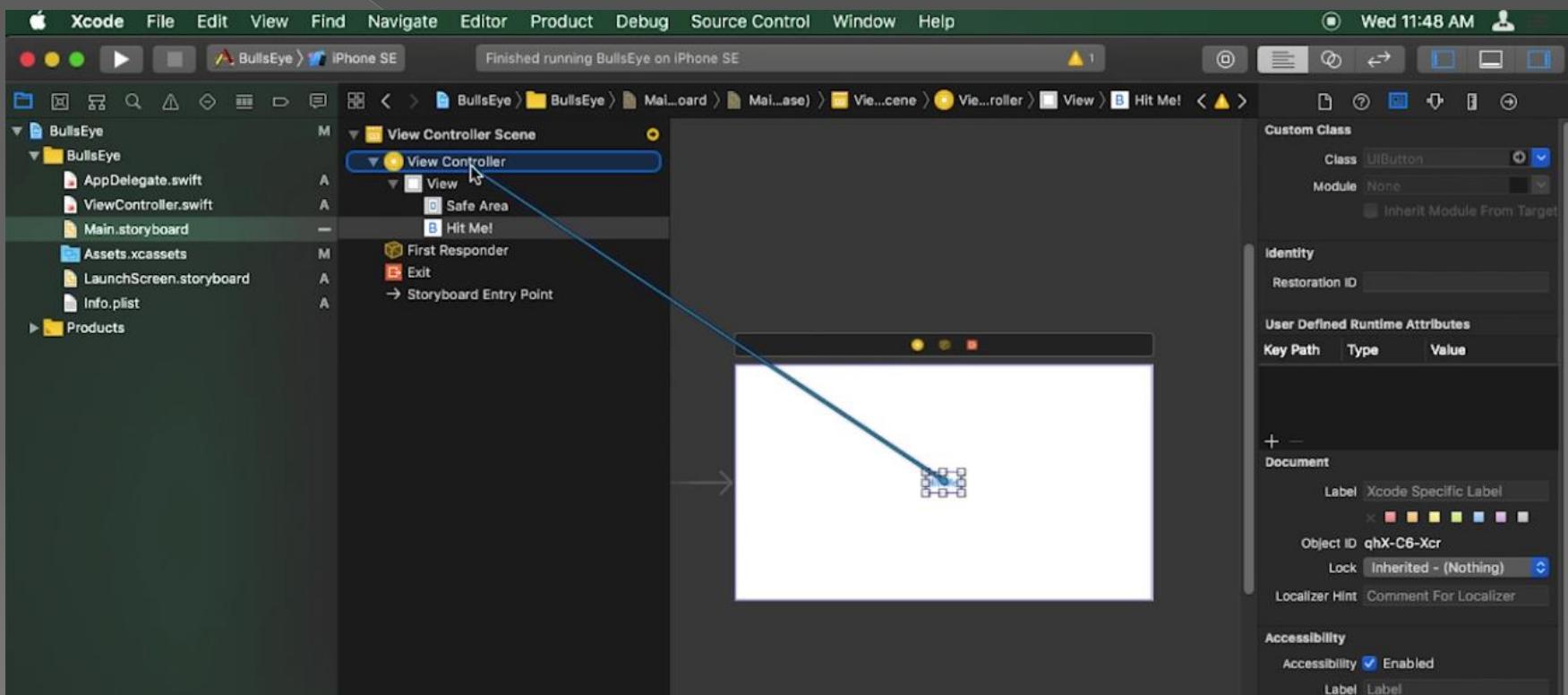
    @IBAction func showAlert() {
        print("Hello!")
    }
}
```

Slika 23 – Dodavanje akcije za button

Povezivanje button-a sa nekom akcijom

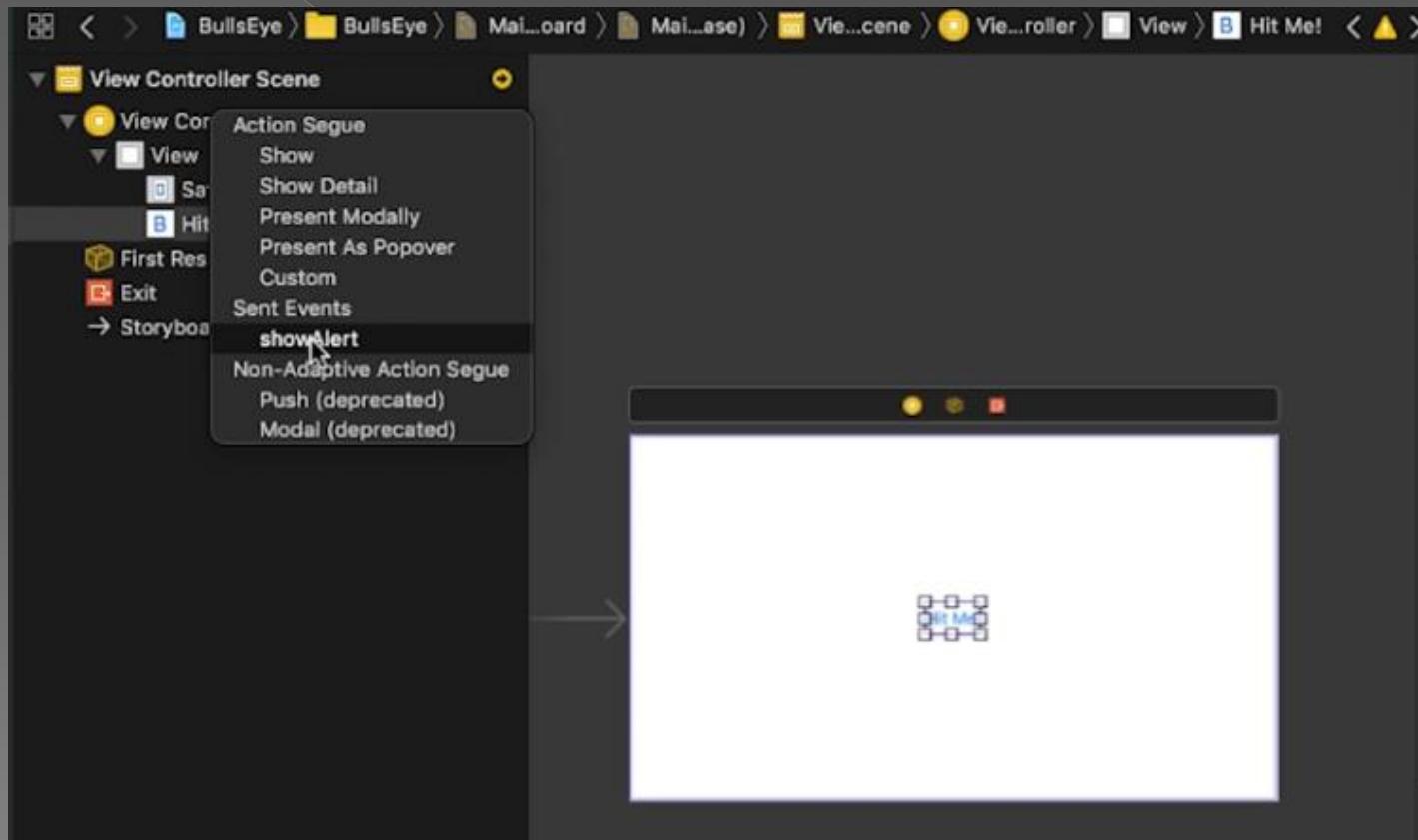
- U ovom primeru, akcija koja treba biti izvršena nakon klika na button je akcija showAlert().
- Povezivanje button-a sa ovom akcijom se vrši jednostavnim fizičkim povezivanjem button-a sa fajlom gde se akcija nalazi.
- Na narednim slikama su prethodno opisani koraci detaljno i lakše pojašnjeni.

Povezivanje button-a sa nekom akcijom



Slika 24 – Povezivanje button-a i akcije

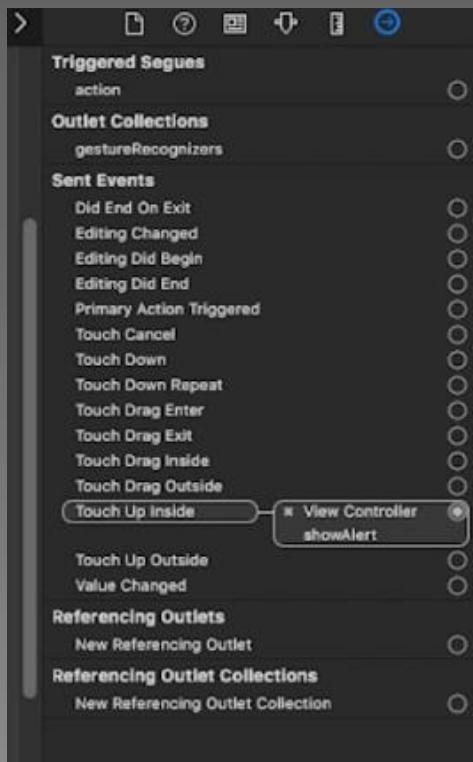
Povezivanje button-a sa nekom akcijom



Slika 25 – Odabir željene akcije

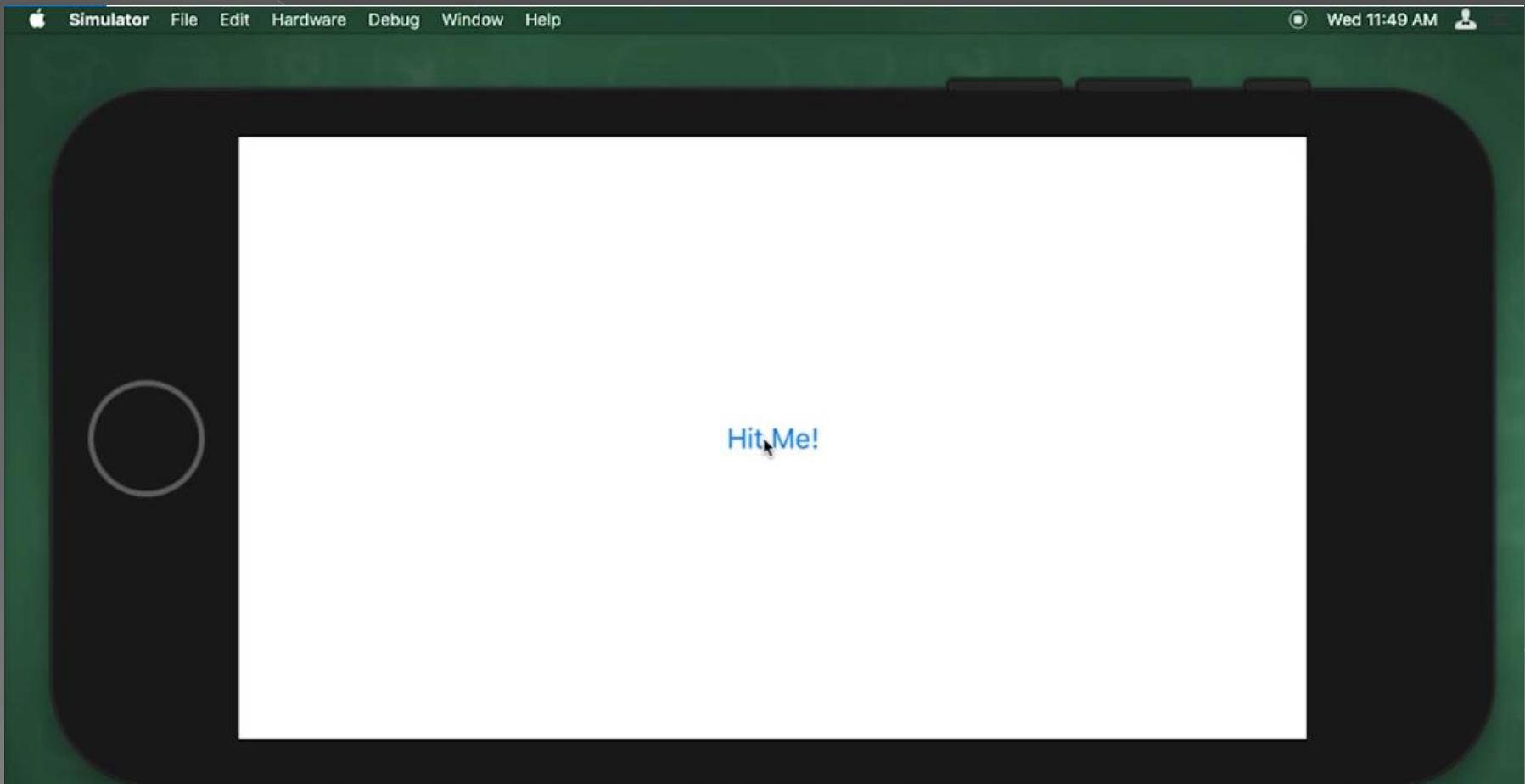
Povezivanje button-a sa nekom akcijom

- Način na koji se vrši provera da li smo ispravno povezali button sa željenom akcijom, pre pokretanja programa je prikazan na slici ispod.
- Naime, u okviru dela gde se vrše podešavanja elemenata aplikacije, možemo videti i njihove osobine koje su im dodeljene kodom.



Slika 26 – Ovde vidimo da je button povezan sa showAlert metodom

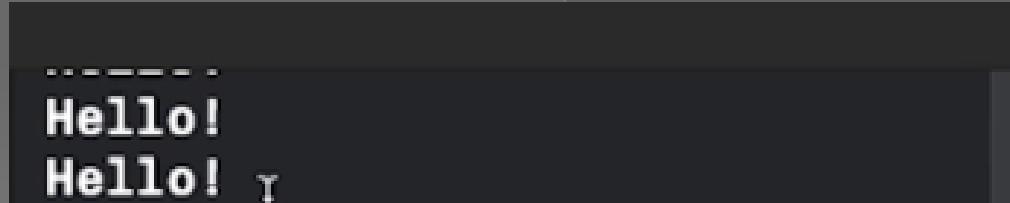
Pokretanje programa iz primera



Slika 27 – Izgled pokrenute aplikacije

Rezultat pokrenutog programa

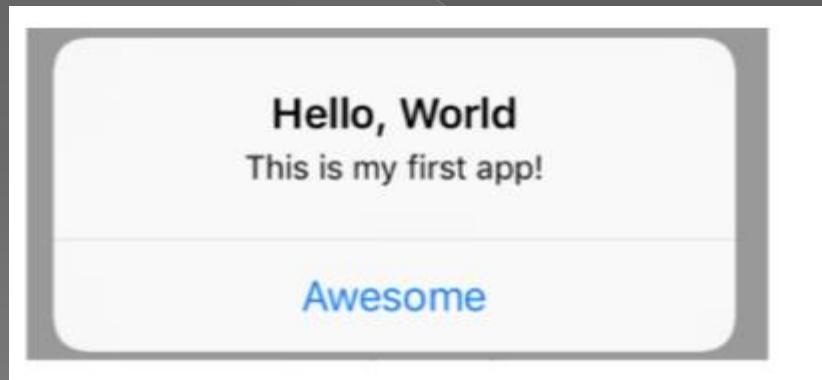
- Na osnovu dodatog koda i akcije koja treba biti izvršena klikom na button, u konzoli treba biti nešto isštampano nakon klika na button.
- Rezultat koji smo mi dobili klikom na button u aplikaciji je baš ovaj koji smo gore naveli, i prikazan je na slici ispod.



Slika 28 – Rezultat pokrenute aplikacije

Pop up poruke

- Sledeći zadatak nam je da klikom na button iz prethodnog primera, nemamo štampanje poruke u konzoli, već da nam se otvori pop up poruka sledećeg izgleda:



Slika 29 – Pop up poruka izgled

Pop up poruke

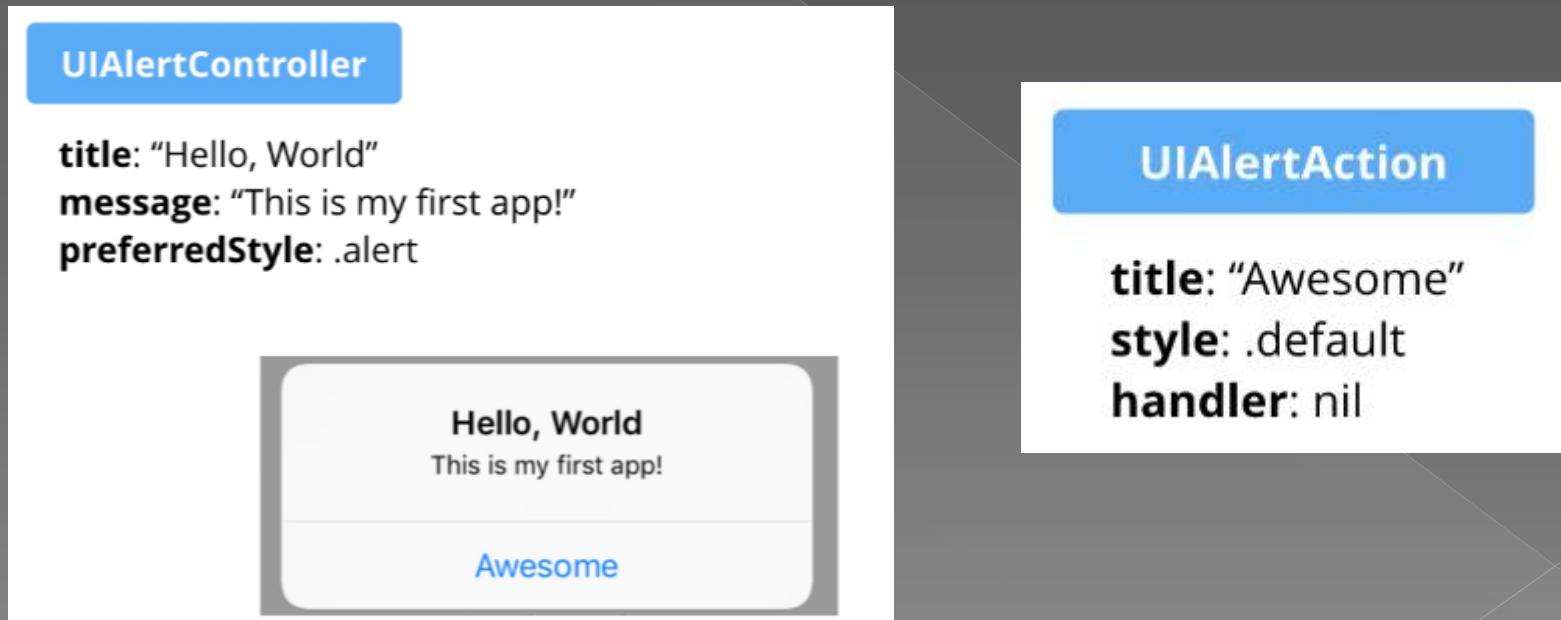
- U nastavku će biti prikazan sadržaj koda aplikacije koji predstavlja pop up poruku.

```
6 // Copyright © 2018 Ray Wenderlich. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController {  
12  
13     override func viewDidLoad() {  
14         super.viewDidLoad()  
15         // Do any additional setup after loading the view, typically from a nib.  
16     }  
17  
18     @IBAction func showAlert() {  
19  
20         let alert = UIAlertController(title: "Hello, World!", message: "This is my  
21             first app!", preferredStyle: .alert)  
22  
23         let action = UIAlertAction(title: "Awesome", style: .default, handler: nil)  
24  
25         alert.addAction(action)  
26  
27         present(alert, animated: true, completion: nil)|  
28 }
```

Slika 30 – Kod za realizovanje Pop up poruke

Pop up poruke

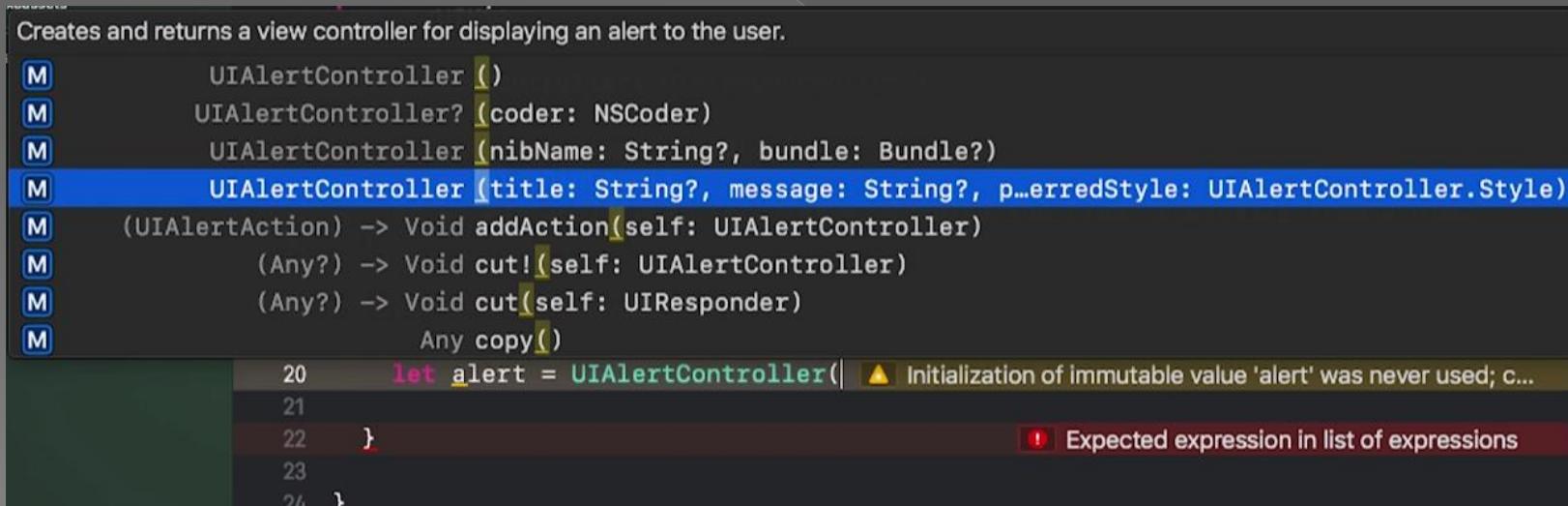
- Kao što se iz priloženog može videti, za kreiranje pop up poruke koriste se dve funkcije `UIAlertController()` i `UIAlertAction` u okviru kojih se definiše i sadržaj dela za naslov za poruku i za dugme. Sve ovo se vrši na osnovu pravila za definisanje ovih sadržaja po sledećem principu:



Slika 31 i 32 – Princip definisanja funkcija

Pop up poruke

- Kao i u svakom programskom jeziku, i u ovom, samo programsko okruženje nam nudi popunu cele funkcije, nakon upisivanja početnih slova na osnovu kojih nam se nudi lista ponuđenih funkcija koje programsko okruženje prepoznaće.
- Ovim se dodatno smanjuje mogućnost pravljenja greške prilikom upotrebe ovih funkcija.



Creates and returns a view controller for displaying an alert to the user.

```
UIAlertController()
UIAlertController?(coder: NSCoder)
UIAlertController(nibName: String?, bundle: Bundle?)
UIAlertController(title: String?, message: String?, preferredStyle: UIAlertController.Style)
(UIAlertAction) -> Void addAction(self: UIAlertController)
(Any?) -> Void cut!(self: UIAlertController)
(Any?) -> Void cut(self: UIResponder)
Any copy()
20 let alert = UIAlertController() // Initialization of immutable value 'alert' was never used; c...
21
22 }
23
24 }
```

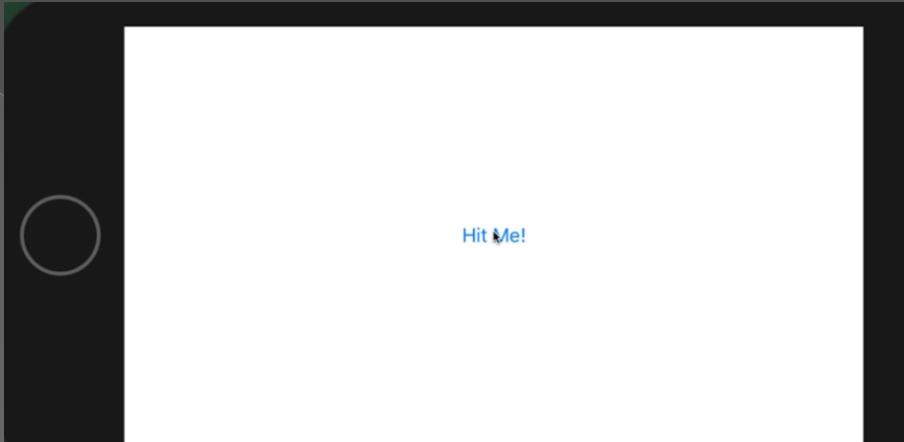
Initialization of immutable value 'alert' was never used; c... Expected expression in list of expressions

Slika 33 – Prepoznavanje funkcije od strane programskog okruženja

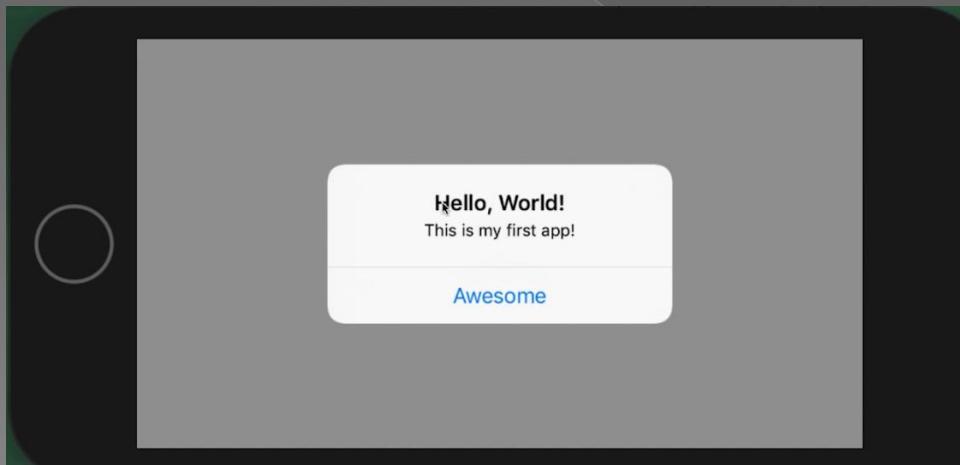
Pop up poruke

- Nakon kompajliranja i startovanja programa, ponovo nam se otvara aplikacija koja izgleda identično aplikaciji prethodnog primera. Razlika ovog i prethodnog primera je upravo pop up poruka. U ovom primeru, nakon klika na button u aplikaciji, otvara nam se pop up poruka koja je definisana na početku razvoja ovog primera. Na slikama u nastavku biće prikazan izgled i funkcionalnost naše nove aplikacije.

Pop up poruke



Slika 34 – Izgled aplikacije



Slika 35 – Pop up poruka koja se štampa nakon klika na button

Debugovanje

- xCode programsko okruženje je jedno od veoma pametnih okruženja, iz razloga što odmah pored greške nudi mogućnost prepravke te greške i jasno ispisuje šta predstavlja tu grešku. U nastavku će biti prikazani primeri sa nekim greškama i ono kako programsko okruženje vidi te greške, prikazuje ih nama i kako nudi rešenje tih grešaka.

Debugovanje

The screenshot shows a Xcode interface with a dark theme. In the center is a code editor displaying Swift code for a ViewController:

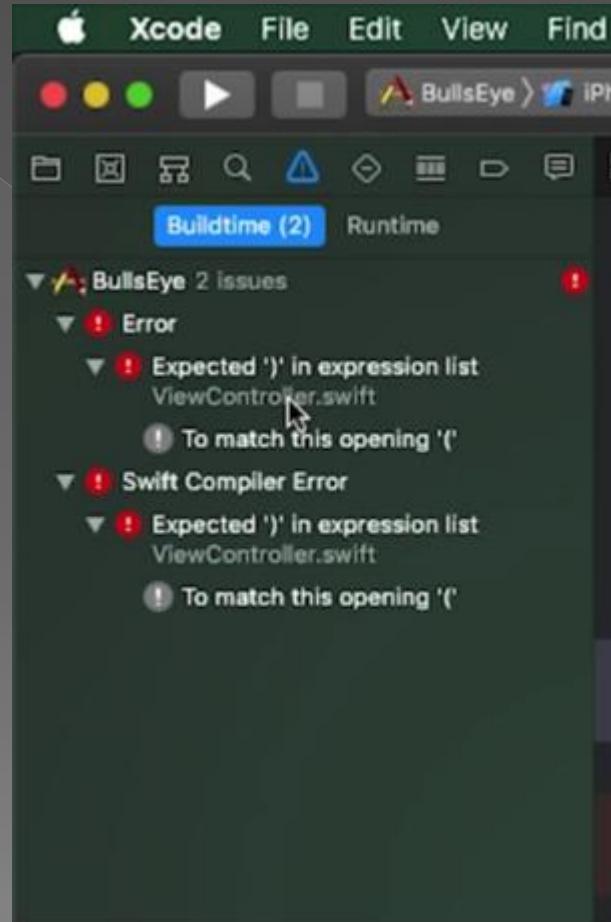
```
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         // Do any additional setup after loading the view, typically from a nib.
16     }
17
18     @IBAction func showAlert() {
19
20         let alert = UIAlertController(title: "Hello, World!", message: "This is my
21             first app!", preferredStyle: .alert)
22
23         let action = UIAlertAction(title: "Awesome", style: .default, handler: nil)
24
25         alert.addAction(action)
26
27         present(alert, animated: true, completion: nil)
28     }
29
30 }
31
32 }
```

A red box highlights the word 'alert' in the line 'let alert = UIAlertController(title: "Hello, World!", message: "This is my first app!", preferredStyle: .alert)'. A dropdown menu appears, listing several suggestions:

- Use of unresolved identifier 'alert'
 - Did you mean 'alart'? Fix
 - Did you mean 'assert'? Fix
 - Did you mean 'abort'? Fix
 - Did you mean 'alarm'? Fix
- Use of unresolved identifier 'alert'
 - Did you mean 'alart'? Fix

Slika 36 – Prikaz greške u programskom okruženju i mogućnost popravke

Debugovanje



Slika 37 – Prikaz još jedne greške koju prepoznaće samo programsko okruženje

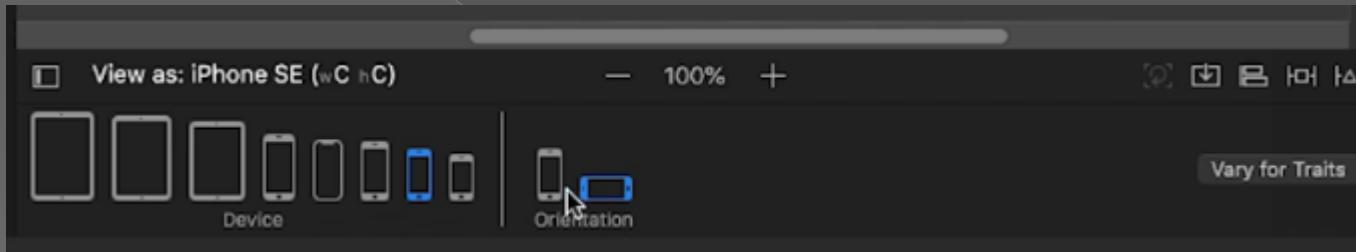
Dimenzije ekrana i layout

- Na slici ispod su prikazane dimenzije ekrana svih Apple uređaja:

Device	Form factor	Screen dimension in points
iPhone 4s and older	3.5"	320 x 480
iPhone 5, 5c, 5s, SE	4"	320 x 568
iPhone 6, 6s, 7, 8	4.7"	375 x 667
iPhone 6, 6s, 7, 8 Plus	5.5"	414 x 736
iPhone X	5.8"	375 x 812
iPad, iPad mini	9.7" and 7.9"	768 x 1024
iPad Pro	10.5"	834 x 1112
iPad Pro	12.9"	1024 x 1366

Slika 38 – Dimenzije ekrana

Dimenzije ekrana i layout



Slika 39 – Biranje orijentacije ekrana

Dimenzije ekrana i layout

- Problem koji može nastati, a vezan je za dimenzije ekrana i layout-a je to da se neka aplikacija lepo prikazuje na primer u horizontalnom prikazu, dok se u vertikalnom prikazu ne prikazuje na celom ekranu ili da se lepo prikazuje na jednom uređaju, da se na drugom ne prikazuje, zbog razlike u dimenzijama.
- Za rešavanje ovog problema koristi se auto layout.
- Definisanje i postavljanje auto layout-a se vrši na sledeći način:

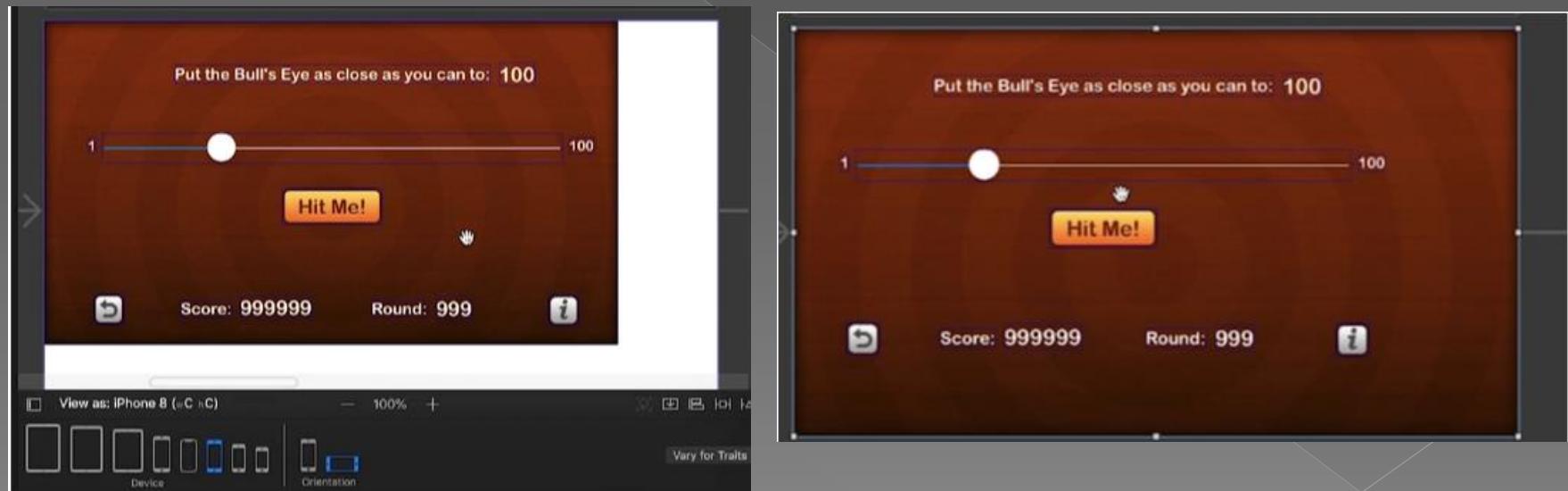
Dimenzijs ekrana i layout



Slika 40 – Izgled dela za podešavanje layout-a, pre i posle postavljanje auto layout - a

Dimenzije ekrana i layout

- Na slikama ispod, kroz primere, biće predstavljeno rešavanje problema prikaza aplikacije na vertikalnom i horizontalnom prikazu.

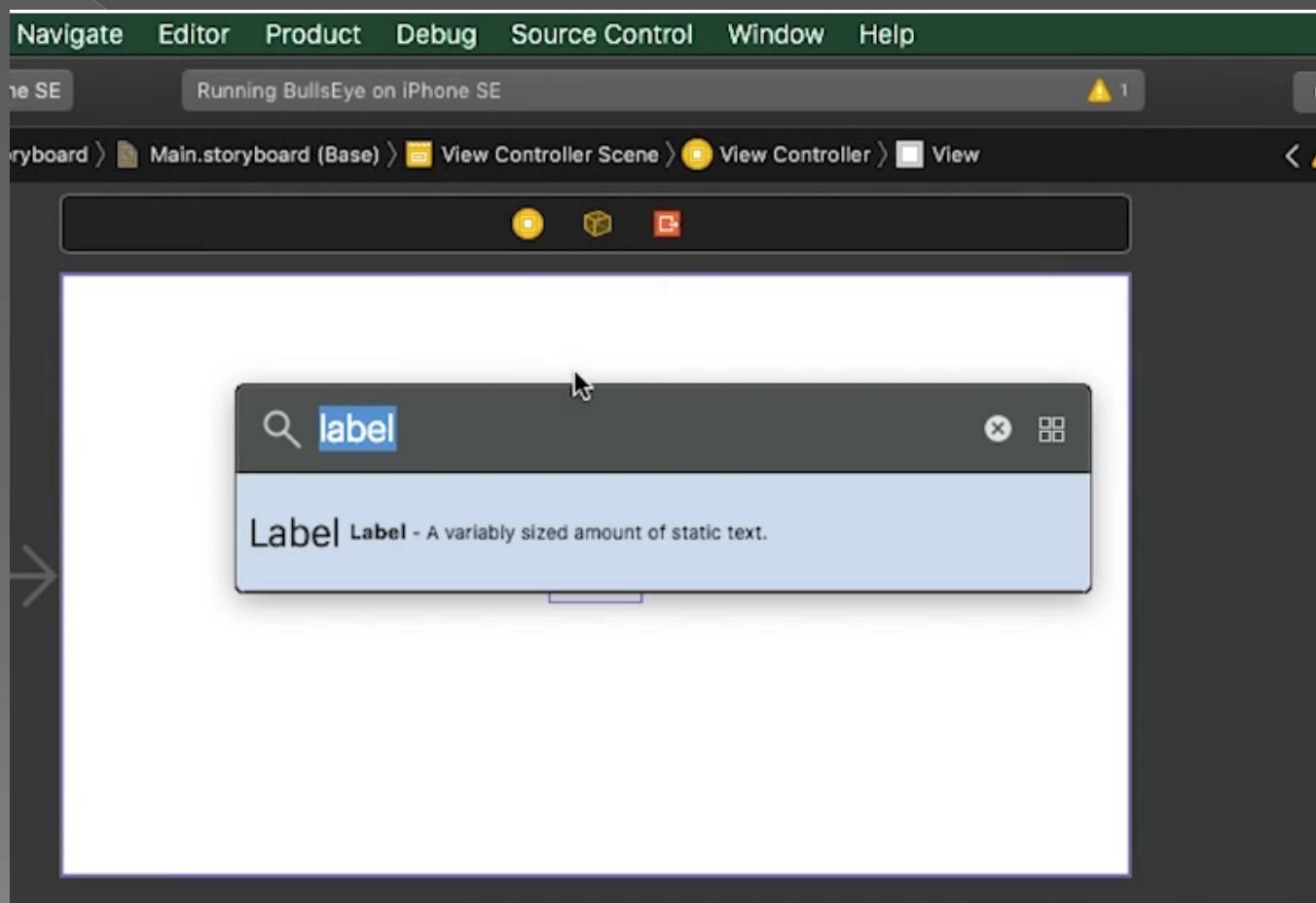


Slika 41 i 42 – Izgled aplikacije pre i posle auto layout-a

Dodavanje teksta

- Dodavanje teksta u prozoru aplikacije se vrši na isti način kao i dodavanje tastera. Prevlačenjem elementa iz liste elemenata koji su dozvoljeni. Na slici ispod je prikazan način dodavanja teksta u aplikaciju.
- Promena teksta se vrši dvoklikom na postojeći tekst.

Dodavanje teksta



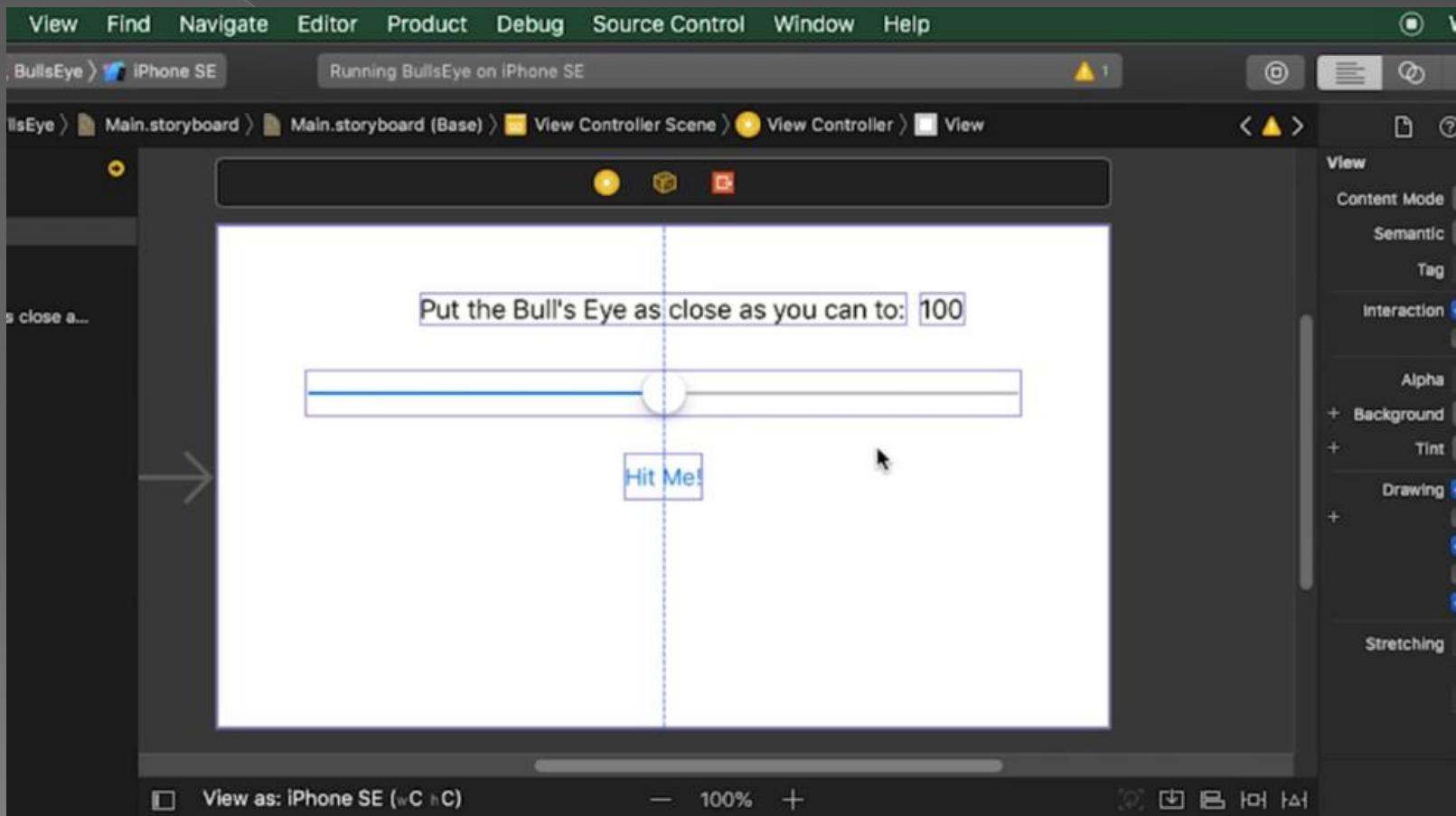
Slika 43 – Proces dodavanja teksta u aplikaciju

Dodavanje slider-a



Slika 44 – Proces dodavanja slider-a

Dodavanje slider-a

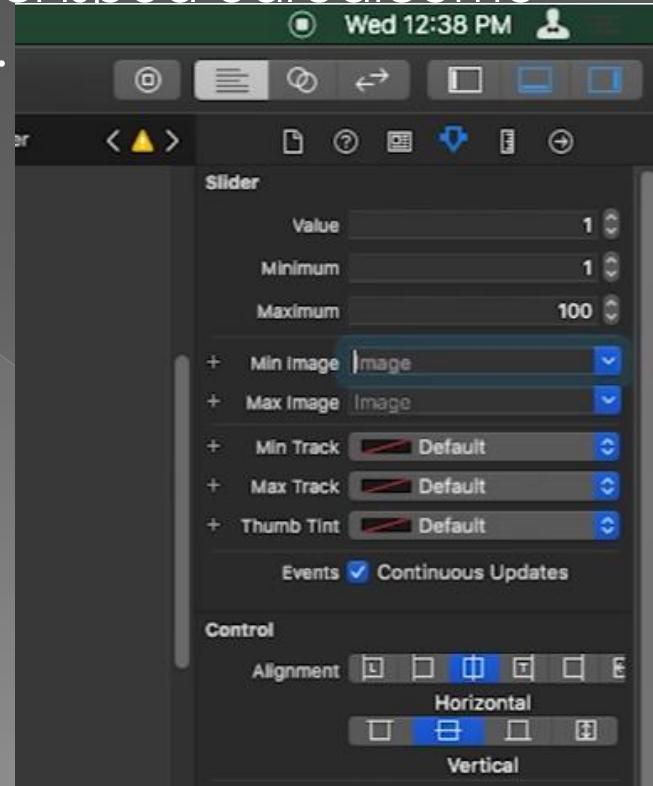


Slika 45 – Proces dodavanja slider-a

Dodavanje slider-a

- Sva podešavanja vezana za osobine slider-a, postavljaju se u okviru padajućeg menija sa slike.
- Uz pomoć ovog padajućeg menija, na slici ispod odredićemo minimalne i maksimalne vrednosti slider-a.

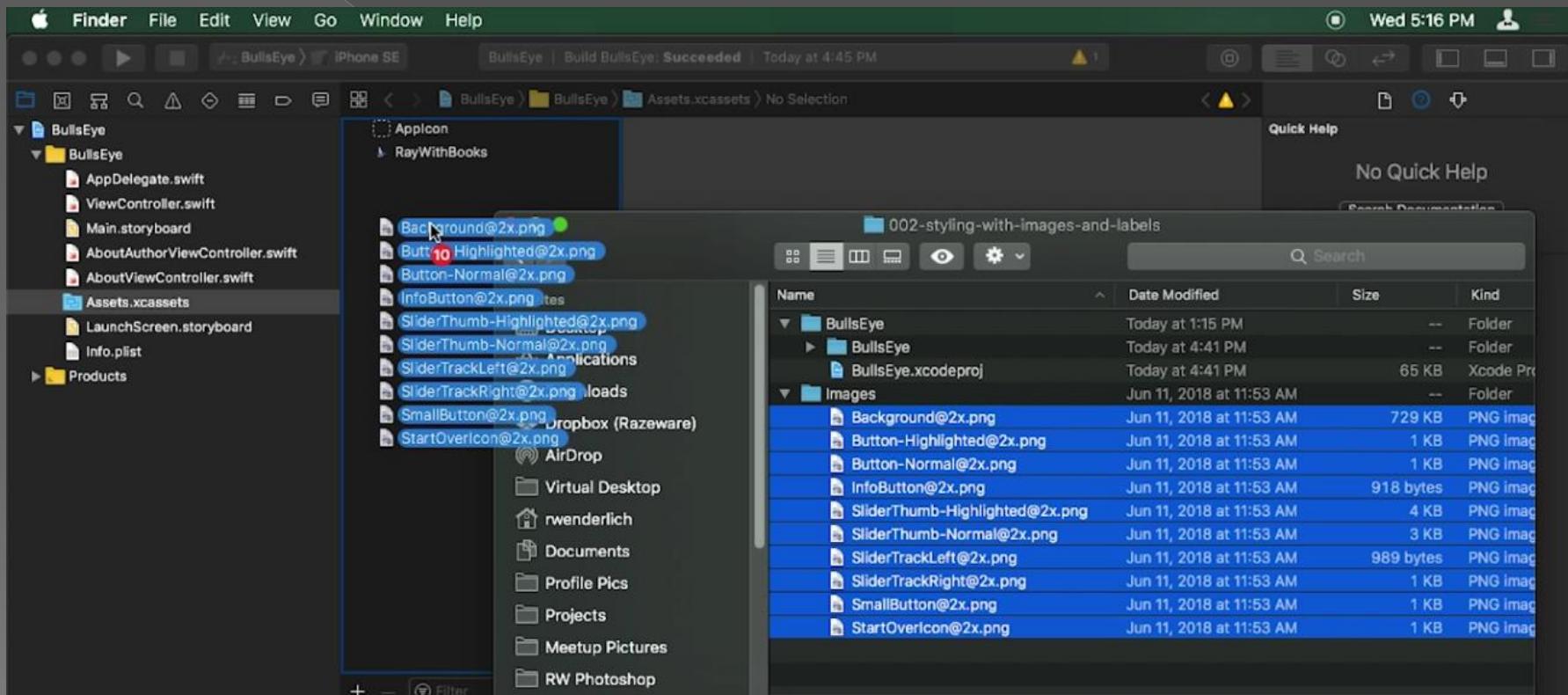
Slika 46 – Podešavanja vrednosti slider-a



Dodavanje slike kao pozadine, button-a i slično

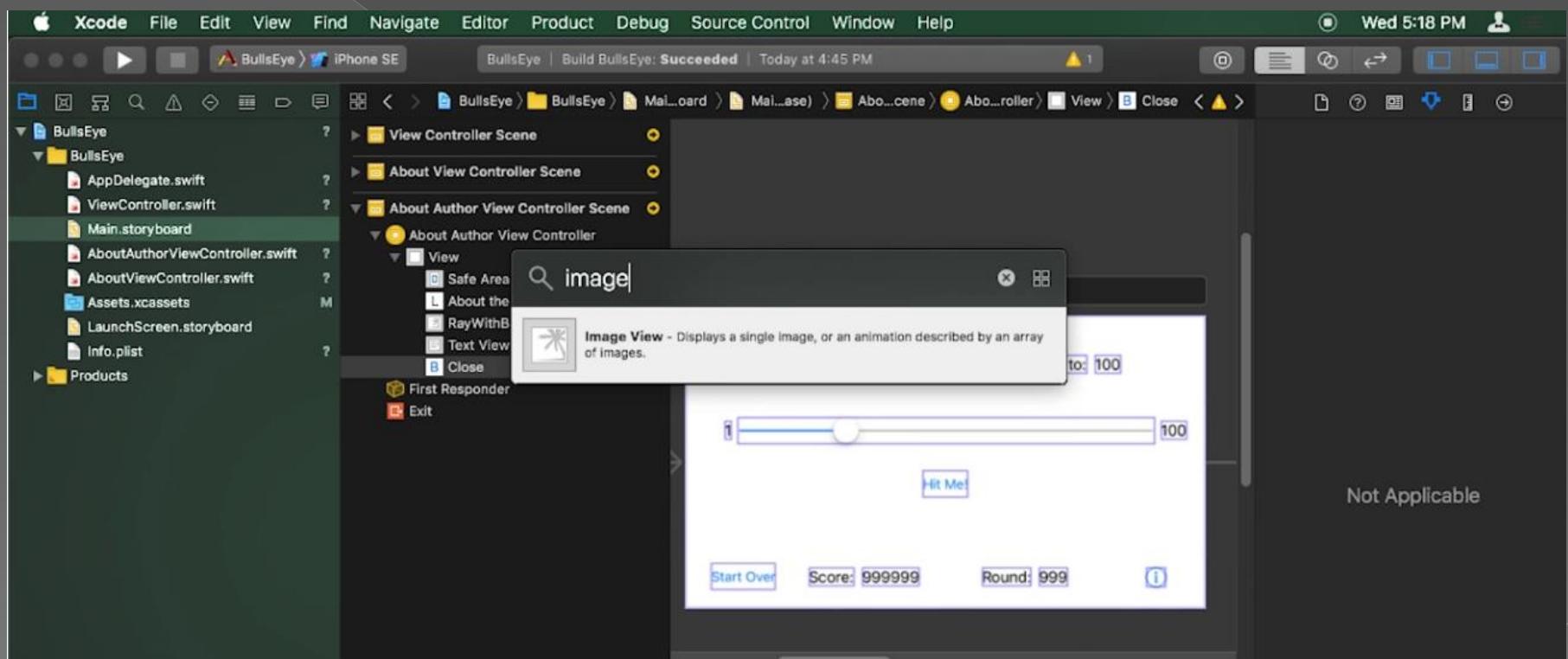
- Prvo što je potrebno da uradimo, kako bismo omogućili kasnije dodavanje slika u program, jeste da slike sa kojima želimo da radimo prekopiramo u fajl projekta xCode programskog okruženja pod nazivom Assets.xcassets.

Dodavanje slike kao pozadine, button-a i slično



Slika 47 – Kopiranje slika u fajl projekta

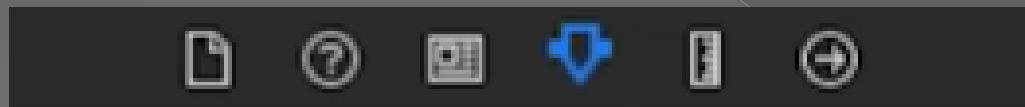
Dodavanje slike kao pozadine, button-a i slično



Slika 48 – Dodavanje slike iz elemenata

Dodavanje slike kao pozadine, button-a i slično

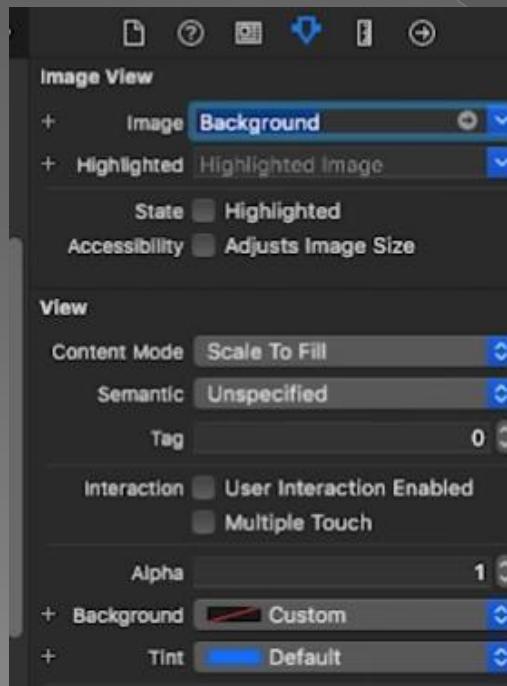
- Da bismo odabrali koju sliku tačno želimo da postavimo u ovo polje koje smo izabrali, prvo moramo otvoriti novi padajući meni koji je strelicom prikazan na slici ispod.



Slika 49 – Padajući meni za podešavanja svih osobina slike

Dodavanje slike kao pozadine, button-a i slično

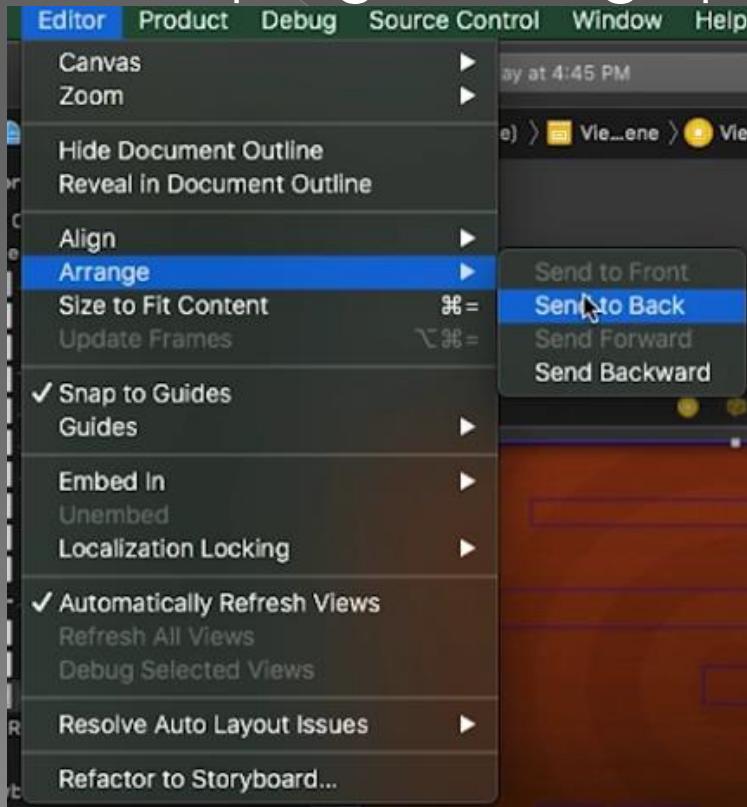
- Da bismo odabrali sliku koju želimo, potrebno je da u polje Image unesemo naziv željene slike, koja je prethodno, po pravilu sačuvana u fajlu projekta.



Slika 50 – Odabir slike

Dodavanje slike kao pozadine, button-a i slično

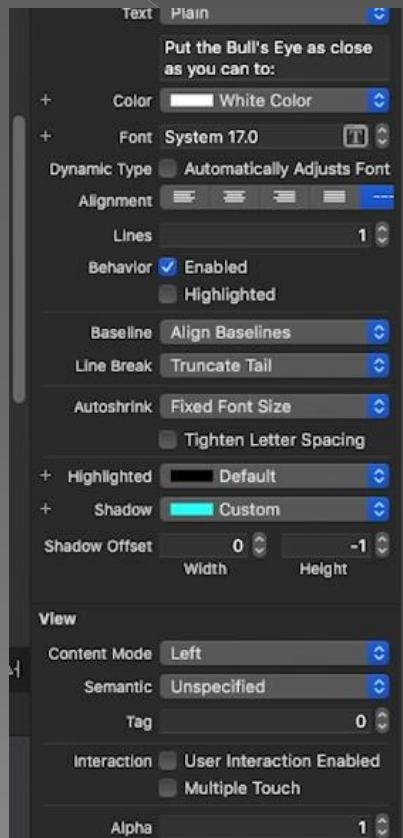
- Podešavanje slike koja je preko teksta, ili ispod teksta, vrši se uz pomoć sledećeg podešavanja u okviru xCode programskog ispita.



Slika 51 – Podešavanje slike preko ili ispod teksta

Podešavanje svojstva teksta

- Sva svojstva teksta, u smislu fonta ,boje, stila itd, postavljaju se u okviru padajućeg menija sa slike ispod:



Slika 52 – Svojstva teksta

Baze podataka i iOS aplikacije

- U nastavku ćemo detaljno objasniti način pravljenja, povezivanja i rada sa bazama podataka.
- Prvi korak u ovom procesu je kreiranje same baze podataka i tabele u okviru te baze podataka.
- Na slici ispod je prikazan kod za pravljenje kao i izgled napravljene tabele.

Baze podataka i iOS aplikacije

- Kreiranje baze podataka se može izvršiti u bilo kom alatu za pravljenje baze podataka, npr. Php myAdmin, Database Manager i slični.
- Može se izvršiti ručno, dodavanjem polja, parametara i elemenata jednog po jednog, ili kodiranjem.
- U nastavku će biti prikazan način pravljenja baze podataka uz pomoć koda, u php myAdmin.
- Nakon kreiranja određene baze podataka, potrebno je izvesti db fajl, u kome će biti sačuvana ta baza podatka, a koji će kasnije biti dodan u programsko okruženje xCode.

Baze podataka i iOS aplikacije

```
1 -- tables
2 -- Table: users
3 CREATE TABLE users (
4     id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
5     username varchar(100) NOT NULL,
6     password text NOT NULL,
7     email varchar(100) NOT NULL,
8     name varchar(100) NOT NULL,
9     phone varchar(10) NOT NULL
10 );
11
12 -- End of file.
```

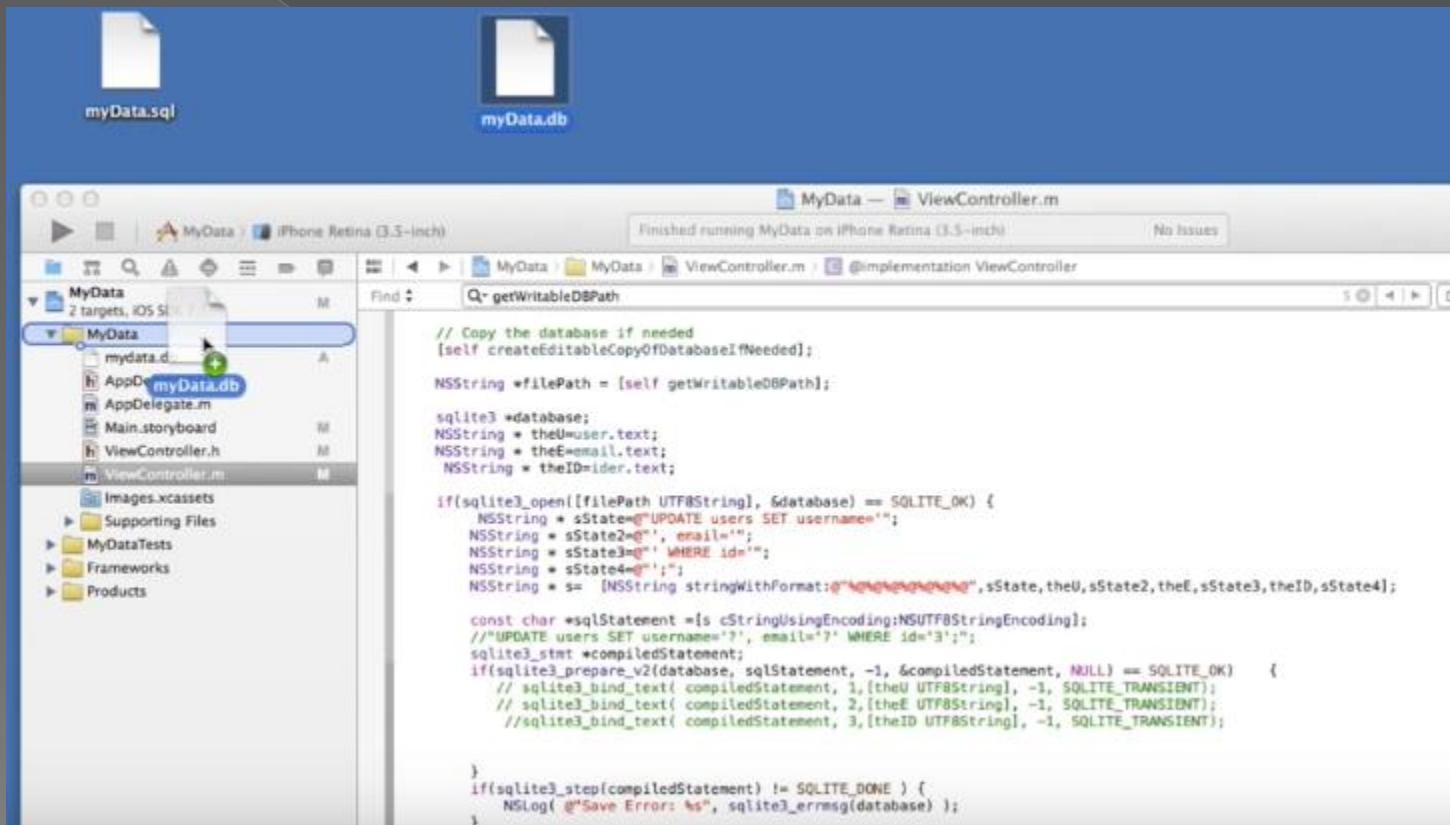
Slika 53 – Kod za kreiranje baze podataka

users		
id	int	PK
username	varchar(100)	
password	text	
email	varchar(100)	
name	varchar(100)	
phone	varchar(10)	

Slika 54 – Kreirana tabela

Baze podataka i iOS aplikacije

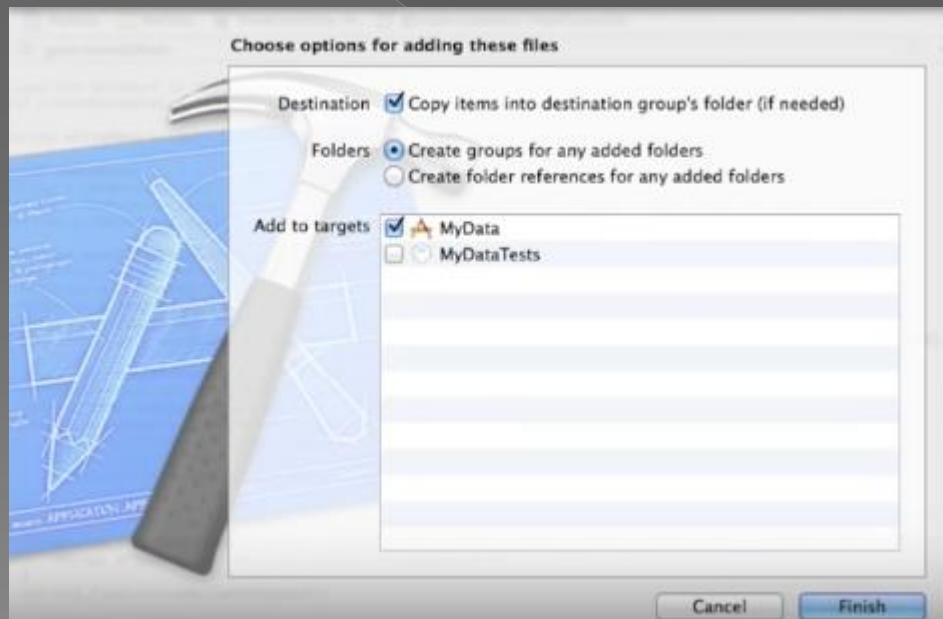
- Kada smo napravili bazu podataka i tabelu u bazi podataka, potrebno je povezati tu bazu podataka sa xCode programom na kome radimo.
- To se radi na vrlo jednostavan način. Naime, uvodimo tu bazu podataka kao eksternu komponentu u programsko okruženje.
- Jednostavnim prevlačenjem napravljene baze podataka u fajl u programskom okruženju xCode.



Slika 55 – Dodavanje baze podataka u programsko okruženje

Baze podataka i iOS aplikacije

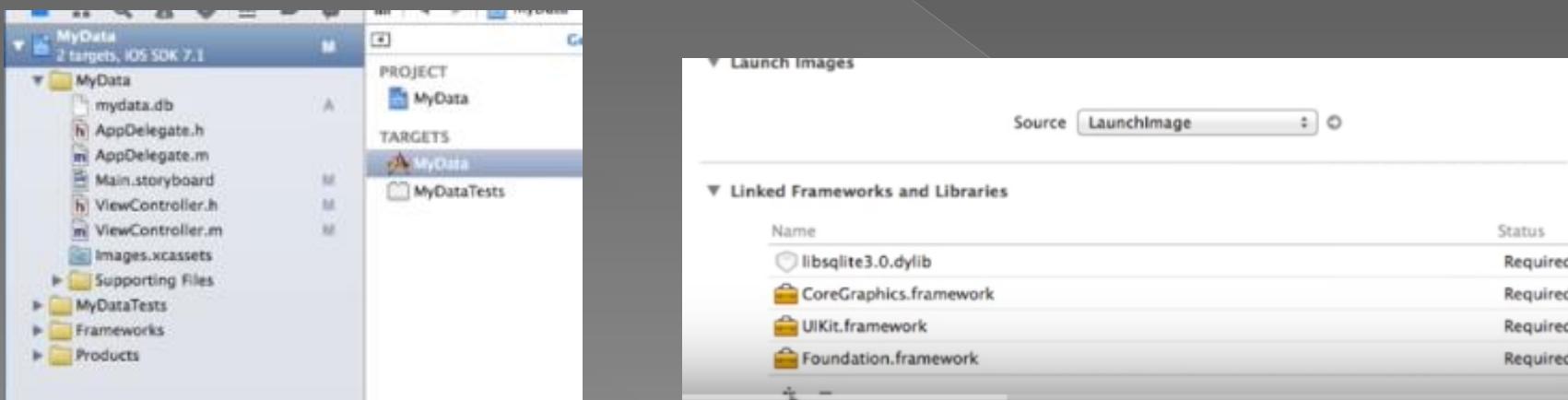
- Nakon prebacivanja fajla baze podataka u odgovarajući fajl u programskom okruženju, otvara nam se prozor u kom biramo kopiranje podataka u dati folder čime završavamo dodavanje baze podataka.



Slika 56 – Dodavanje baze podataka

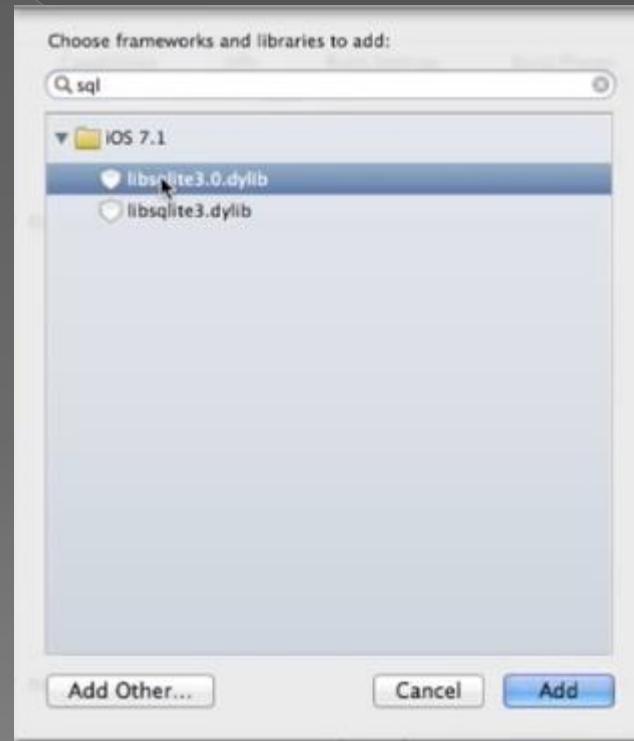
Baze podataka i iOS aplikacije

- Sledeći korak je povezivanje napravljene baze podataka sa kodom u xCode programskom okruženju. Ovo se vrši na sledeći način:
- Kliknuti dva puta na MyData, skrolati otvoreno polje do dna gde piše Linked Frameworks and Libraries, a zatim kliknuti na znak plus.
- Kada se otvori polje za biranje biblioteka u polje za pretragu upisati sql i dodati prvu ponuđenu biblioteku.



Slika 57 – Dodavanje SQL biblioteke

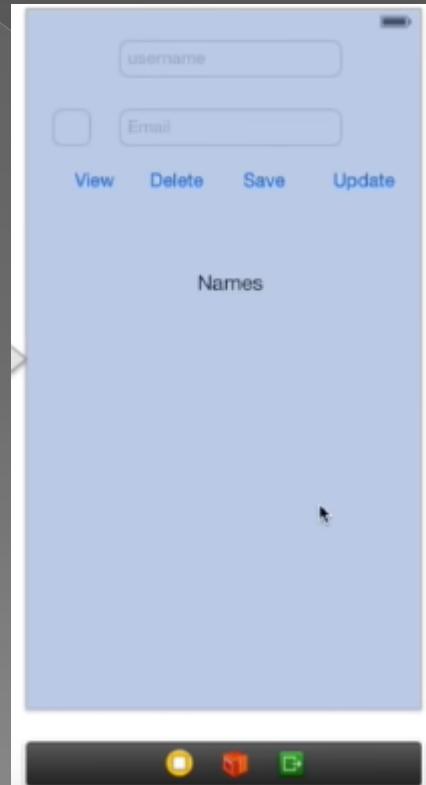
Baze podataka i iOS aplikacije



Slika 58 – Dodavanje sql biblioteke

Baze podataka i iOS aplikacije

- Izgled aplikacije koju ćemo koristiti za prikaz i objašnjenje rada sa bazom podataka je prikazan na slici ispod:



Slika 59 – Izgled aplikacije

Baze podataka i iOS aplikacije

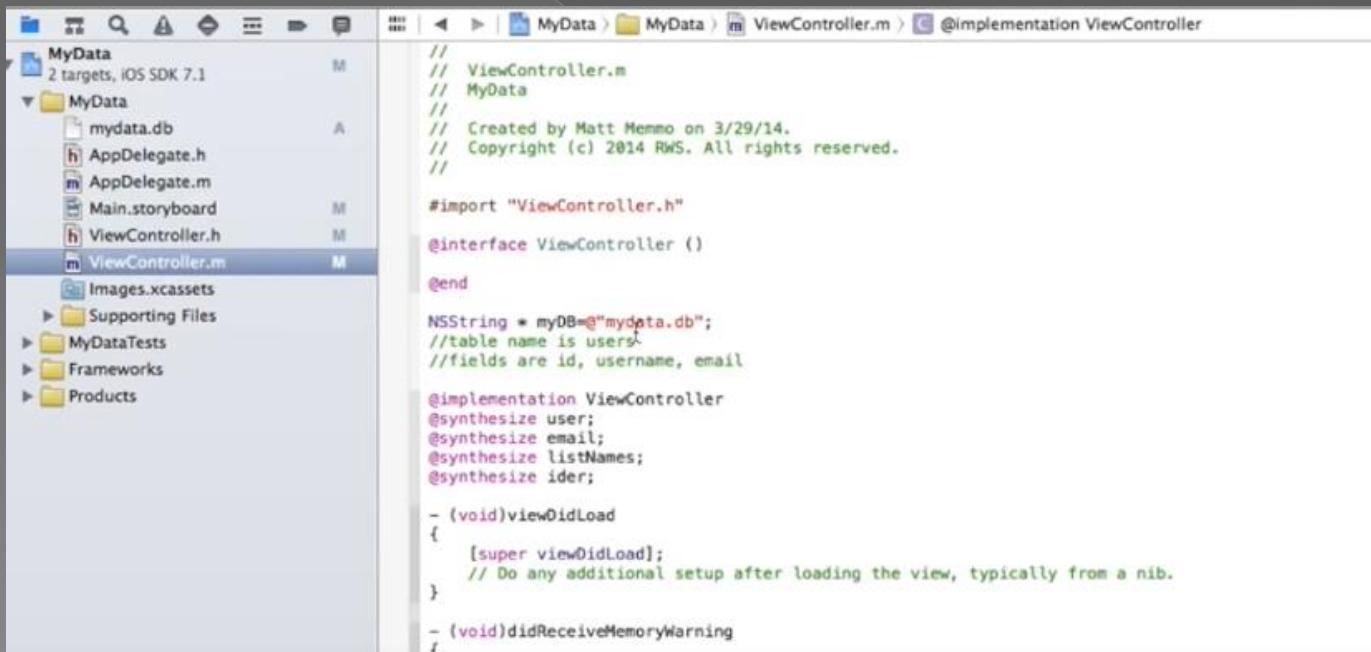
- Ono što je bitno napomenuti je to da u fajlu u kome se nalazi programski kod je neophodno importovati sqlite.h fajl, koji nam omogućava rad sa sql bazama podataka i daje mogućnost upotrebe svih funkcija vezanih za baze podataka.

```
//  
// ViewController.h  
// MyData  
//  
// Created by Matt Memmo on 3/29/14.  
// Copyright (c) 2014 RWs. All rights reserved.  
  
#import <UIKit/UIKit.h>  
#import <sqlite3.h>  
  
@interface ViewController : UIViewController  
  
@property (weak, nonatomic) IBOutlet UITextField *user;  
  
@property (weak, nonatomic) IBOutlet UITextField *email;  
  
- (IBAction)view:(id)sender;  
- (IBAction)deleteName:(id)sender;  
  
- (IBAction)addName:(id)sender;  
- (IBAction)update:(id)sender;  
@property (weak, nonatomic) IBOutlet UILabel *listNames;
```

Slika 60 – Kod

Baze podataka i iOS aplikacije

- Povezivanje koda aplikacije sa bazom podataka vrši se kodiranjem, na način prikazan na slici ispod. Bitno je da budemo sigurni da smo u kodu napisali naziv baze podataka koju smo prethodno importovali u naše programsko okruženje.



The screenshot shows the Xcode interface with the project 'MyData' selected. The left sidebar displays the file structure:

- MyData (2 targets, iOS SDK 7.1)
 - MyData
 - mydata.db
 - AppDelegate.h
 - AppDelegate.m
 - Main.storyboard
 - ViewController.h
 - ViewController.m** (selected)
 - Images.xcassets
 - Supporting Files
 - MyDataTests
 - Frameworks
 - Products

The right pane shows the code for `ViewController.m`:

```
// Viewcontroller.m
// MyData
//
// Created by Matt Memmo on 3/29/14.
// Copyright (c) 2014 RWS. All rights reserved.

#import "ViewController.h"

@interface ViewController : UIViewController

@end

NSString *myDB=@"mydata.db";
//table name is users
//fields are id, username, email

@implementation ViewController
@synthesize user;
@synthesize email;
@synthesize listNames;
@synthesize ider;

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning
{
}
```

Slika 61 – Povezivanje sa bazom podataka

Baze podataka i iOS aplikacije

- Kod za prikaz podataka iz baze podataka je prikazan na slici ispod:

```
- (IBAction)view:(id)sender {//================================================================VIEW=====

    NSString* emailN;
    NSString * theID;
    NSString* userN;
    NSString * myLine=@" ";
    NSString * paths=[self getWritableDatabasePath];

    const char *dbpath = [paths UTF8String];
    sqlite3_stmt *statement;
    static sqlite3 *database = nil;

    if (sqlite3_open(dbpath, &database) == SQLITE_OK)
    {
        NSString *querySQL = [NSString stringWithFormat:@"SELECT username, email, id FROM users",nil];
        const char *query_stmt = [querySQL UTF8String];
        // NSLog(@"Databasae opened = %@", userN);

        if (sqlite3_prepare_v2(database,
                              query_stmt, -1, &statement, NULL) == SQLITE_OK)
        {
            int rows = sqlite3_column_int(statement, 0);
            while(sqlite3_step(statement) == SQLITE_ROW)
            {
                userN = [[NSString alloc]initWithUTF8String:(const char *) sqlite3_column_text(statement, 0)];
                emailN = [[NSString alloc]initWithUTF8String:(const char *) sqlite3_column_text(statement, 1)];
                theID = [[NSString alloc]initWithUTF8String:(const char *) sqlite3_column_text(statement, 2)];
                myLine= [NSString stringWithFormat:@"%@%@",myLine,userN,theID,@" "];

            }
            // username.text=@"No Username";

            sqlite3_finalize(statement);
        }
        // NSLog( @"Save Error: %s", sqlite3_errmsg(database) );
        sqlite3_close(database);
    }
    // NSLog(@"user = %@", userN);
    listNames.text=myLine;
}
```



Slika 62 – Prikaz elemenata iz baze podataka 72

Baze podataka i iOS aplikacije

- Brisanje elemenata iz baze podataka.

```
- (IBAction)deleteName:(id)sender {//=================================================================DELETE=====
    // Copy the database if needed
    [self createEditableCopyOfDatabaseIfNeeded];

    NSString *filePath = [self getWritableDBPath];

    sqlite3 *database;
    NSString * theID=ider.text;

    if(sqlite3_open([filePath UTF8String], &database) == SQLITE_OK) {

        NSString * sstate=@"DELETE FROM users WHERE id=''";
        NSString * s= [NSString stringWithFormat:@"%@%@\%@",sstate,theID,@""];
        const char *sqlstatement = [s cStringUsingEncoding:NSUTF8StringEncoding];//"DELETE FROM users WHERE id='3'";
        sqlite3_stmt *compiledStatement;
        sqlite3_prepare_v2(database, sqlstatement, -1, &compiledStatement, NULL);

        if(sqlite3_step(compiledStatement) != SQLITE_DONE ) {
            NSLog( @"Save Error: %@", sqlite3_errmsg(database) );
        }
        sqlite3_finalize(compiledStatement);
    }
    sqlite3_close(database);
}

- (IBAction)addName:(id)sender {
    [self saveUserInDatabase];
}
```

Baze podataka i iOS aplikacije

○ Ažuriranje elemenata u bazi podataka:

```
- (IBAction)update:(id)sender {//=====================================================================
    // Copy the database if needed
    [self createEditableCopyOfDatabaseIfNeeded];

    NSString *filePath = [self getWritableDBPath];

    sqlite3 *database;
    NSString * theU=user.text;
    NSString * theE=email.text;
    NSString * theID=ider.text;

    if(sqlite3_open([filePath UTF8String], &database) == SQLITE_OK) {
        NSString * sState=@"UPDATE users SET username='";
        NSString * sState2=@", email='";
        NSString * sState3=@"' WHERE id='";
        NSString * sState4=@"';";
        NSString * s= [NSString stringWithFormat:@"%@%@%@%@%@",sState,theU,sState2,theE,sState3,theID,sState4];

        const char *sqlStatement =[s cStringUsingEncoding:NSUTF8StringEncoding];
        //@"UPDATE users SET username='?', email='?' WHERE id='3'";
        sqlite3_stmt *compiledStatement;
        if(sqlite3_prepare_v2(database, sqlStatement, -1, &compiledStatement, NULL) == SQLITE_OK) {
            // sqlite3_bind_text( compiledStatement, 1,[theU UTF8String], -1, SQLITE_TRANSIENT);
            // sqlite3_bind_text( compiledStatement, 2,[theE UTF8String], -1, SQLITE_TRANSIENT);
            //sqlite3_bind_text( compiledStatement, 3,[theID UTF8String], -1, SQLITE_TRANSIENT);

        }
        if(sqlite3_step(compiledStatement) != SQLITE_DONE ) {
            NSLog( @"Save Error: %@", sqlite3_errmsg(database) );
        }
        sqlite3_finalize(compiledStatement);
    }
    sqlite3_close(database);
    // UPDATE users SET username='?', email='?' WHERE id='1';
}
```

Slika 64 - Ažuriranje

Baze podataka i iOS aplikacije

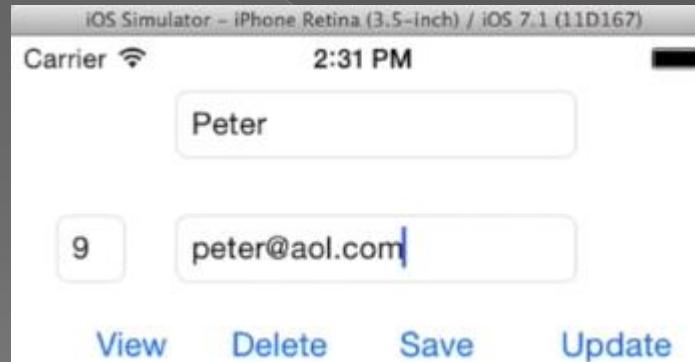
- Čuvanje elemenata unetih iz forme aplikacije u bazu podataka:

```
- (void)saveUserInDatabase { //=====SAVE=====  
    // Copy the database if needed  
    [self createEditableCopyOfDatabaseIfNeeded];  
  
    NSString *filePath = [self getWritableDatabasePath];  
  
    sqlite3 *database;  
    NSString * theU=user.text;  
    NSString * theE=email.text;  
  
    if(sqlite3_open([filePath UTF8String], &database) == SQLITE_OK) {  
        //NSString *temp = [NSString stringWithFormat:@"insert into allusers (user_id,user_name) VALUES (%@,%@)",user_id,user_name];  
        const char *sqlStatement = "insert into users (username,email) VALUES (?,?)";  
        sqlite3_stmt *compiledStatement;  
        if(sqlite3_prepare_v2(database, sqlStatement, -1, &compiledStatement, NULL) == SQLITE_OK) {  
            sqlite3_bind_text( compiledStatement, 1,[theU UTF8String], -1, SQLITE_TRANSIENT);  
            sqlite3_bind_text( compiledStatement, 2,[theE UTF8String], -1, SQLITE_TRANSIENT);  
  
        }  
        if(sqlite3_step(compiledStatement) != SQLITE_DONE ) {  
            NSLog( @"Save Error: %@", sqlite3_errmsg(database) );  
        }  
        sqlite3_finalize(compiledStatement);  
    }  
    sqlite3_close(database);  
}
```

Slika 65 – Čuvanje unetih elemenata

Startovanje programa

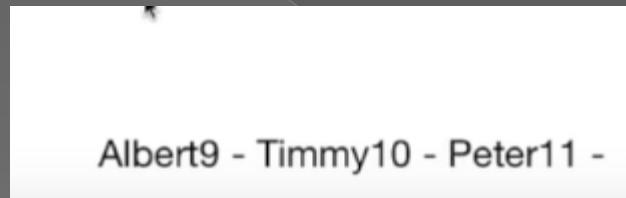
- Prilikom startovanja prethodno razvijene aplikacije, dobijamo izgled ekrana prikazan na slici ispod.



Slika 66 – Pokretanje aplikacije

Startovanje programa

- Nakon unošenja podataka u polja aplikacije i klika na dugme save, unutar baze podataka dodaju se uneti parametri.



Slika 67 – Prikaz elemenata baze podataka

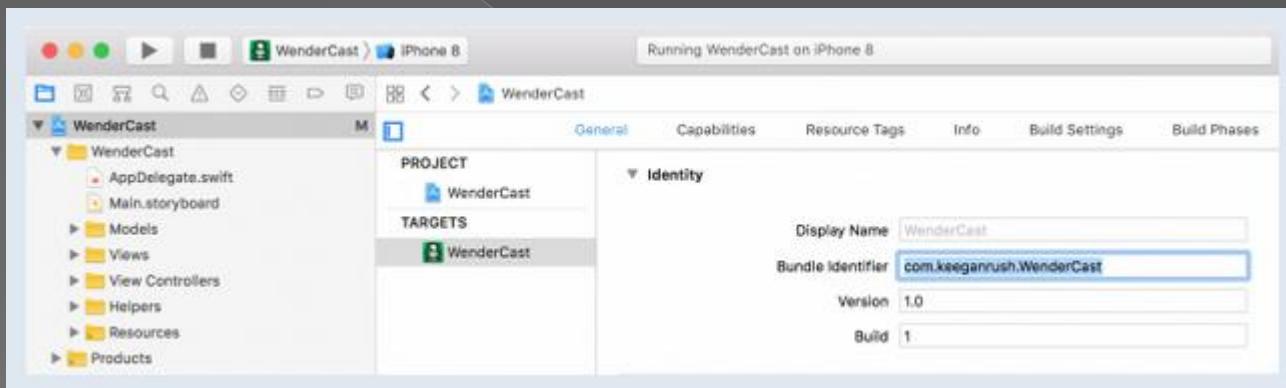
Notifications

- iOS programeri vole da konstantno zamišljaju korisnike koji koriste svoju sjajnu aplikaciju. Naravno, korisnici će ponekad morati da zatvore aplikaciju i da obavljaju druge aktivnosti. Na sreću, push obaveštenja omogućavaju programerima da dođu do korisnika i obavljaju male zadatke, čak i kada korisnici aktivno ne koriste aplikaciju.
- Slanje push obaveštenja je odgovornost serverske komponente vaše aplikacije. Mnoge aplikacije koriste treće strane za slanje push obaveštenja, dok druge koriste prilagođena rešenja ili popularne biblioteke (npr. Houston).

Notifications

- Bezbednost je veoma važna za push obaveštenja, budući da ne želimo da bilo ko drugi šalje naša obaveštenja putem aplikacija. Potrebno je da izvršimo nekoliko zadataka da bismo konfigurisali aplikaciju da bezbedno primi push obaveštenja.
- Prvo treba promeniti ID aplikacije. U xCode programskom okruženju postavimo naš projekat na vrh Project Navigatora, a zatim selektujemo target našeg projekta. Otvorimo Generals i zatim promenimo Bundle Identifier na nešto tako da Apple-ov server za push obaveštenja može usmeriti pritisak na ovu aplikaciju.

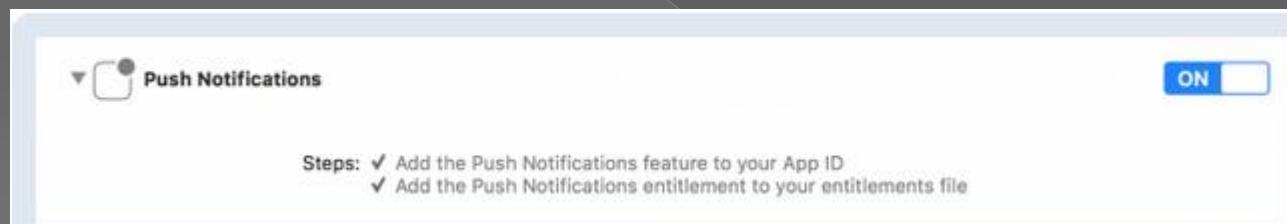
Notifications



Slika 68 – Promena identifikatora

Notifications

- Zatim moramo da kreiramo ID aplikacije na svom nalogu i omogućimo pravo na push obaveštenja. xCode ima jednostavan način da ovo uradimo. Kliknimo na karticu Capabilities i postavimo prekidač push notifications-a na ON.



Slika 69 – Dozvola push notifications dodatka

Notifications

- Postoje dva koraka za registraciju za push obaveštenja. Prvo, moramo dobiti dozvolu korisnika za prikazivanje obaveštenja. Zatim, možemo da registrujemo uređaj da bismo primali udaljena push obaveštenja. Ako sve prođe dobro, sistem će nam omogućiti device token, koji možemo zamisliti kao adresu za ovaj uređaj.

Notifications

- Sada otvorimo AppDelegate.swift i dodajmo prvo na samom vrhu koda importovanje UserNotifications-a, a zatim na kraju dodati sledeću funkciju:

```
import UserNotifications
```

Slika 70 – Dodavanje na početku koda

```
func registerForPushNotifications() {
    UNUserNotificationCenter.current() // 1
        .requestAuthorization(options: [.alert, .sound, .badge]) { // 2
            granted, error in
                print("Permission granted: \(granted)") // 3
        }
}
```

Slika 71 – Dodavanje na kraju koda

Notifications

- UNUserNotificationCenter - obrađuje sve aktivnosti vezane za obaveštavanja u aplikaciji;
- Pozivamo `requestAuthorization(options:completionHandler:)` na zahtev za autorizaciju za prikazivanje obaveštenja. Prošle opcije ukazuju na vrste obaveštenja koje želimo da aplikacija koristi – ovde zahtevamo obaveštenje, zvuk i oznaku.
- Rukovalac završetka prima Bool vrednost koja pokazuje da li je autorizacija bila uspešna.

Notifications

- Zatim dodajemo sledeće blizu kraja aplikacije:

```
registerForPushNotifications()
```

Pozivanje ove funkcije obezbeđuje da će aplikacija pokušati da se registruje za push obaveštenja u bilo kom trenutku kada se pokrene.

Notifications

- Nakon kompajliranja i startovanja aplikacije, trebalo bi da dobijemo prompt koji traži dozvolu za slanje obaveštenja i koji izgleda kao na slici ispod:



Slika 72 – Pokretanje dosadašnje aplikacije

Notifications

- Potrebno je odobriti, kako bi aplikacija sada mogla prikazivati obaveštenja.
- Da bi omogućili korisniku da izabere opciju don't allow potrebno je dodati sledeću metodu unutar AppDelegate fajla.

```
func getNotificationSettings() {
    UNUserNotificationCenter.current().getNotificationSettings { settings in
        print("Notification settings: \(settings)")
    }
}
```

Slika 73 – Notifications dodatak

Notifications

- Dodavanjem `requestAuthorization(options:completionHandler:)` funkcije u `getNotificationSettings` fajl, omogućeno je korisniku da može, u bilo kom trenutku, otići u postavke aplikacije i promeniti svoje dozvole za obaveštavanje.

```
UNUserNotificationCenter.current()
    .requestAuthorization(options: [.alert, .sound, .badge]) {
    [weak self] granted, error in

        print("Permission granted: \(granted)")
        guard granted else { return }
        self?.getNotificationSettings()
}
```

Slika 74 – Omogućavanje promene dozvola

Notifications

- Sada kada imamo dozvole, sada ćemo se registrovati za udaljena obaveštenja.
- U getNotificationSettings() dodajemo sledeće:

```
guard settings.authorizationStatus == .authorized else { return }
DispatchQueue.main.async {
    UIApplication.shared.registerForRemoteNotifications()
}
```

Slika 75 – Dodavanje dozvole za udaljena obaveštenja

Notifications

- Dodajemo sledeće dve metode na kraj aplikacije AppDelegate-a. iOS će nas pozvati da nas obavesti o rezultatu.

```
func application(
    _ application: UIApplication,
    didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data
) {
    let tokenParts = deviceToken.map { data in String(format: "%02.2hhx", data) }
    let token = tokenParts.joined()
    print("Device Token: \(token)")
}

func application(
    _ application: UIApplication,
    didFailToRegisterForRemoteNotificationsWithError error: Error) {
    print("Failed to register: \(error)")
}
```

Slika 76 – Dodavanje koda

Notifications

- Da biste mogli da pošaljete push obaveštenje morate da posetite centar za članove Apple developer-a i da se prijavite.
- U centru za članove, izaberite Certificates, Identifiers & Profiles, a zatim Keys. U gornjem desnom uglu kliknite na dugme + da biste kreirali novi ključ.

Notifications

The screenshot shows the 'Certificates, Identifiers & Profiles' section of the Apple Developer portal. On the left, a sidebar lists categories: Certificates, Keys (selected), Identifiers, Devices, and a dropdown for iOS, tvOS, watchOS. Under Keys, 'All' is selected. In the main area, a 'Create New Key' dialog is open. It has a title bar 'Create New Key' with a '+' and search icon. Below it is a sub-section titled 'Create a New Key'. A 'Key Description' section contains a 'Name:' field with 'Push Notification Key' and a note about special characters. A 'Key Services' section contains a 'Service' button and a list where 'Apple Push Notifications service (APNs)' is checked. A descriptive text explains that APNs establishes connectivity between the notification server and the Apple Push Notification service.

Certificates, Identifiers & Profiles

iOS, tvOS, watchOS

Keegan Rush

Certificates

- All
- Pending
- Development
- Production

Keys

- All

Identifiers

- App IDs
- Pass Type IDs
- Website Push IDs
- iCloud Containers
- App Groups
- Merchant IDs
- Music IDs
- Maps IDs

Devices

- All
- Apple TV

Create New Key

Create and configure services for this key.

Create a New Key

Key Description

Name: Push Notification Key
You cannot use special characters such as @, &, ., ;, *

Key Services

Enable Service

Apple Push Notifications service (APNs)
Establish connectivity between your notification server and the Apple Push Notification service. One key is used for all of your apps. [Learn more](#)

Slika 77 – Kreiranje ključa

Notifications

- ◉ Potrebno je preuzeti PushNotifications program sa sledećeg linka:

<https://github.com/onmyway133/PushNotifications>

Notifications

- Nakon preuzimanja Push Notifications fajla, potrebno je pokrenuti ga i završiti sledeće korake:
 - Nakon Autentifikacije, odaberite Token;
 - Kliknite na Select P8 dugme i odaberite .p8 fajl iz prethodnog odabira;
 - Upišite vaš Key ID i Team ID u relevantna polja.

Notifications

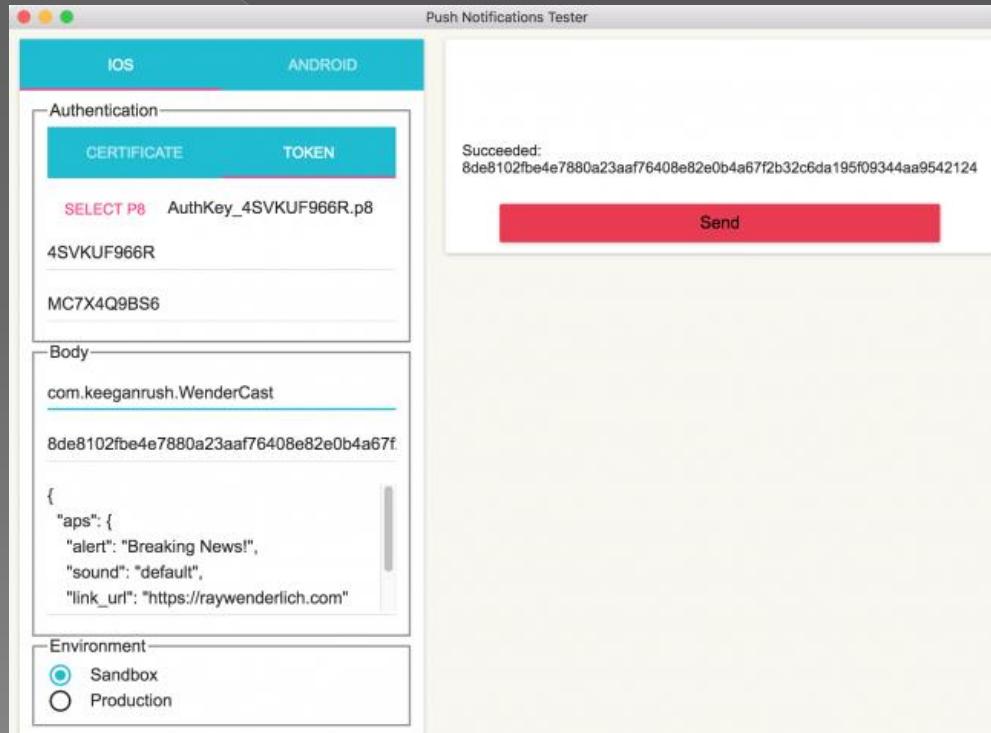
- Promenite request body u sledeći izgled:

```
{  
    "aps": {  
        "alert": "Breaking News!",  
        "sound": "default",  
        "link_url": "https://raywenderlich.com"  
    }  
}
```

- Kliknite na Send button u Push Notifications programu.

Notifications

- Izgled aplikacije i prikaz odabira obaveštenja koje se šalje:



Slika 78 – Izgled aplikacije za slanje obaveštenja

Notifications

- Klikom na dugme send u aplikaciji sa prethodne slike, trebalo bi da ste primili poruku na Vašem uređaju. U ovom primeru, poruka na našem uređaju je primljena i izgled te poruke je prikazan na slici ispod:



Slika 79 – Izgled ekrana našeg uređaja