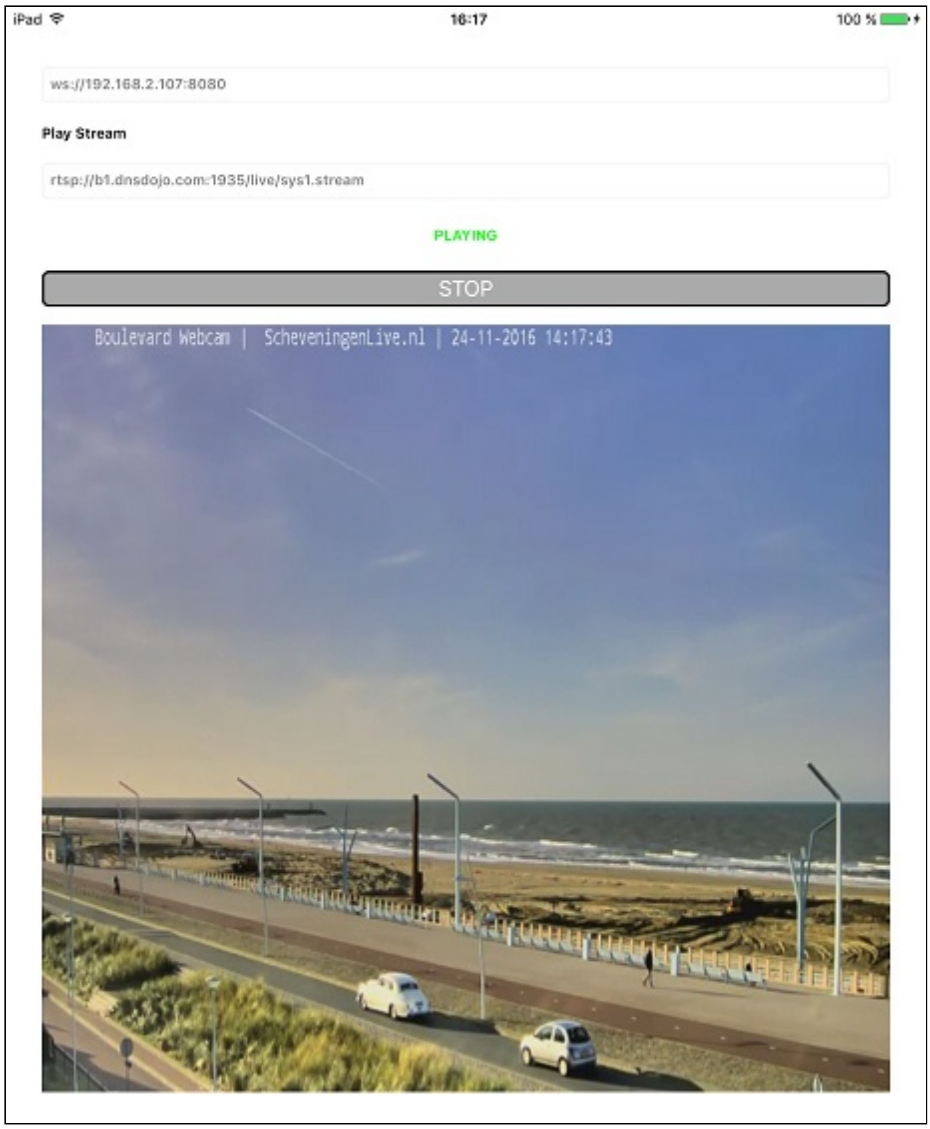
iOS Player

Пример плеера для iOS

Данный плеер может использоваться для воспроизведения любого типа потока с Web Call Server:

- RTSP
- WebRTC
- RTMP

На скриншоте ниже представлен пример во время воспроизведения RTSP-потока.
В URL в поле ввода `WCS URL` `192.168.2.107` - адрес WCS-сервера.
В поле ввода `Play Stream` - имя потока, в данном случае RTSP URL.



Работа с кодом примера

Для разбора кода возьмем версию примера Player, которая доступна здесь.

Класс для основного вида приложения: `ViewController` (заголовочный файл ViewController.h; файл имплементации ViewController.m).

1. Импорт API

code

```
#import <FPWCSApi2/FPWCSApi2.h>
```

2. Создание сессии

`FPWCSApi2.createSession` code

В параметрах сессии указываются:

- URL WCS-сервера
- имя серверного приложения `defaultApp`

```
FPWCSApi2SessionOptions *options = [[FPWCSApi2SessionOptions alloc] init];
options.urlServer = _connectUrl.text;
options.appKey = @"defaultApp";
NSError *error;
FPWCSApi2Session *session = [FPWCSApi2 createSession:options error:&error];
```

3. Подключение к серверу

`FPWCSApi2Session.connect` code

```
[session connect];
```

4. Получение от сервера события, подтверждающего успешное соединение

`FPWCSApi2Session.onConnected` code

При получении данного события вызывается метод воспроизведения потока `ViewController playStream`

```
- (void)onConnected:(FPWCSApi2Session *)session {
    [self changeViewState:_remoteStreamName enabled:NO];
    [self playStream];
}
```

5. Воспроизведение видеопотока

`FPWCSApi2Session.createStream`, `FPWCSApi2Stream.play` code

Методу `createStream` передаются параметры:

- имя воспроизводимого потока
- вид для отображения потока

```
- (FPWCSApi2Stream *)playStream {
    FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = _remoteStreamName.text;
    options.display = _remoteDisplay;
    NSError *error;
```

```
    FPWCSApi2Stream *stream = [session createStream:options error:nil];
    ...
    if(![stream play:&error]) {
        UIAlertController * alert = [UIAlertController
                                      alertControllerWithTitle:@"Failed to
play"
                                      message:error.localizedDescription

preferredStyle:UIAlertControllerStyleAlert];

        UIAlertAction* okButton = [UIAlertAction
                                    actionWithTitle:@"Ok"
                                    style:UIAlertActionStyleDefault
                                    handler:^(UIAlertAction * action) {

                                    }];

        [alert addAction:okButton];
        [self presentViewController:alert animated:YES completion:nil];
    }
    return stream;
}
```

6. Закрытие соединения

`FPWCSApi2Session.disconnect` code

```
- (void)startButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        if ([FPWCSApi2 getSessions].count) {
            FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
            NSLog(@"Disconnect session with server %@", [session
getServerUrl]);
            [session disconnect];
        } else {
            NSLog(@"Nothing to disconnect");
            [self onDisconnected];
        }
    } else {
        [self changeViewState:_connectUrl enabled:NO];
        [self connect];
    }
}
```

7. Получение события, подтверждающего разъединение

`FPWCSApi2Session.onDisconnected` code

```
- (void)onDisconnected {
    [self changeViewState:_connectUrl enabled:YES];
    [self onStopped];
}
```