iOS Video Chat

Пример iOS-приложения для видеочата

Данный пример может использоваться для участия в видеочате для двух пользователей на Web Call Server и позволяет публиковать WebRTC-поток.

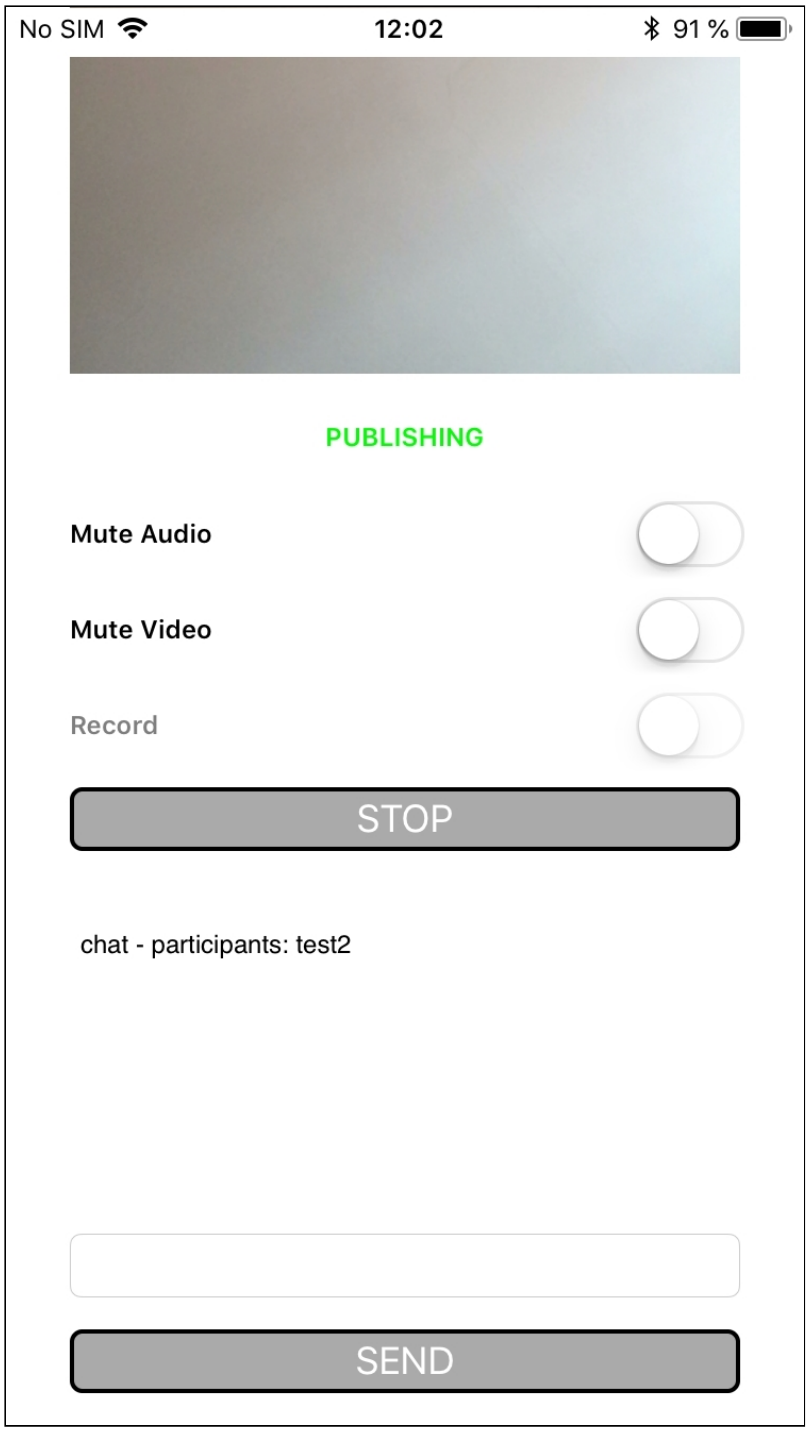
На скриншоте ниже представлен пример с комнатой, к которой присоединился другой участник.

Поля ввода, необходимые для установления соединения и присоединения к комнате

- `WCS URL` - адрес WCS-сервера
- `Login` - имя пользователя
- `Room` - имя комнаты чата

На скриншоте вопроизводятся два видео

- нижнее - видео с камеры данного участника
- верхнее - видео от другого участника



Работа с кодом примера

Для разбора кода возьмем версию примера Video Chat, 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. Подключение к серверу

`FPWCSApi2.createRoomManager` code

В параметрах сессии указываются:

- URL WCS-сервера
- имя пользователя чат-комнаты

```
- (void)connect {
    FPWCSApi2RoomManagerOptions *options = [[FPWCSApi2RoomManagerOptions
alloc] init];
    options.urlServer = _connectUrl.text;
    options.username = _connectLogin.input.text;
    NSError *error;
    roomManager = [FPWCSApi2 createRoomManager:options error:&error];
    ...
}
```

3. Присоединение к конференции

`FPWCSApi2RoomManager.join` code

Методу передаются параметры:

- имя чат-комнаты

```
FPWCSApi2RoomOptions * options = [[FPWCSApi2RoomOptions alloc] init];
options.name = _joinRoomName.input.text;
room = [roomManager join:options];
```

4. Получение от сервера события, подтверждающего успешное присоединение к конференции

`FPWCSApi2Room.onStateCallback` code

При получении данного события:

- количество и состав других участников определяется с помощью метода `FPWCSApi2Room.getParticipants`
- если количество участников более 2, текущий участник выходит из комнаты

- если текущий участник остается в комнате, запускается проигрывание потока от других участников при помощи метода `FPWCSApi2RoomParticipant.play`

```
[room onStateCallback:^(FPWCSApi2Room *room) {
    NSDictionary *participants = [room getParticipants];
    if ([participants count] >= 2) {
        [room leave:nil];
        _joinStatus.text = @"Room is full";
        [self changeViewState:_joinButton enabled:YES];
        return;

    }
    NSString *chatState = @"participants: ";
    for (NSString* key in participants) {
        FPWCSApi2RoomParticipant *participant = [participants
valueForKey:key];
        ParticipantView *pv = [freeViews pop];
        [busyViews setValue:pv forKey:[participant getName]];
        [participant play:pv.display];
        pv.login.text = [participant getName];
        chatState = [NSString stringWithFormat:@"%@%@, ", chatState,
[participant getName]];
    }
    ...
}];
```

5. Публикация видеопотока

`FPWCSApi2Room.publish` code

Методу передаются параметры:

- вид для локального отображения публикуемого потока
- `record` определяет, необходимо ли записывать видеопоток при публикации

```
- (void)publishButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        [room unpublish];
    } else {
        FPWCSApi2StreamOptions * options = [[FPWCSApi2StreamOptions alloc]
init];
        options.record = [_record.control isOn];
        publishStream = [room publish:_localDisplay withOptions:options];
        ...
    }
}
```

6. Получение от сервера события, сигнализирующего о присоединении к конференции другого участника

`FPWCSApi2Room.kFPWCSRoomParticipantEventJoined` code

```
[room on:kFPWCSRoomParticipantEventJoined participantCallback:^(FPWCSApi2Room
*room, FPWCSApi2RoomParticipant *participant) {
    ParticipantView *pv = [freeViews pop];
    if (pv) {
        pv.login.text = [participant getName];
        _messageHistory.text = [NSString stringWithFormat:@"%@\n%@ - %@",
_messageHistory.text, participant.getName, @"joined"];
        [busyViews setValue:pv forKey:[participant getName]];
    }
}];
```

7. Получение от сервера события, сигнализирующего о публикации видеопотока другим участником, и воспроизведение видеопотока.

`FPWCSApi2Room.kFPWCSRoomParticipantEventPublished`,
`FPWCSApi2RoomParticipant.play` code

```
[room on:kFPWCSRoomParticipantEventPublished
participantCallback:^(FPWCSApi2Room *room, FPWCSApi2RoomParticipant
*participant) {
    ParticipantView *pv = [busyViews valueForKey:[participant getName]];
    if (pv) {
        [participant play:pv.display];
    }
}];
```

8. Получение от сервера события, сигнализирующего о получении сообщения от другого участника.

`FPWCSApi2Room.onMessageCallback` code

```
[room onMessageCallback:^(FPWCSApi2Room *room, FPWCSApi2RoomMessage *message)
{
    _messageHistory.text = [NSString stringWithFormat:@"%@\n%@ - %@",
_messageHistory.text, message.from, message.text];
}];
```

9. Отправка текстового сообщения

`FPWCSApi2RoomParticipant.sendMessage` code

Методу передается текст сообщения.

```
- (void)sendButton:(UIButton *)button {
    for (NSString *name in [room getParticipants]) {
        FPWCSApi2RoomParticipant *participant = [room getParticipants][name];
```

```
        [participant sendMessage:_messageBody.text];
    }
    _messageHistory.text = [NSString stringWithFormat:@"%@\n%@ - %@",
_messageHistory.text, _connectLogin.input.text, _messageBody.text];
    _messageBody.text = @"";
}
```

10. Включение/выключение аудио и видео для публикуемого потока

`FPWCSApi2Stream.unmuteAudio`, `FPWCSApi2Stream.muteAudio`,
`FPWCSApi2Stream.unmuteVideo`, `FPWCSApi2Stream.muteVideo` code

```
- (void)muteAudioChanged:(id)sender {
    if (publishStream) {
        if (_muteAudio.control.isOn) {
            [publishStream muteAudio];
        } else {
            [publishStream unmuteAudio];
        }
    }
}

- (void)muteVideoChanged:(id)sender {
    if (publishStream) {
        if (_muteVideo.control.isOn) {
            [publishStream muteVideo];
        } else {
            [publishStream unmuteVideo];
        }
    }
}
```

11. Остановка публикации видеопотока

`FPWCSApi2Room.unpublish` code

```
- (void)publishButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        [room unpublish];
    } else {
        ...
    }
}
```

12. Выход из комнаты конференции

`FPWCSApi2Room.leave` код

Методу передается обработчик ответа REST hook-приложения WCS-сервера.

```
if ([button.titleLabel.text isEqualToString:@"LEAVE"]) {
    if (room) {
        FPWCSApi2DataHandler *handler = [[FPWCSApi2DataHandler alloc] init];
        handler.onAccepted = ^(FPWCSApi2Session *session, FPWCSApi2Data
*data){
            [self onUnpublished];
            [self onLeaved];
        };
        handler.onRejected = ^(FPWCSApi2Session *session, FPWCSApi2Data
*data){
            [self onUnpublished];
            [self onLeaved];
        };
        [room leave:handler];
        room = nil;
    }
}
```

13. Закрытие соединения

`FPWCSApi2RoomManager.disconnect` code

```
- (void)connectButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"DISCONNECT"]) {
        if (roomManager) {
            [roomManager disconnect];
        }
        ...
    }
}
```