Workshop: Telegram Bot

—

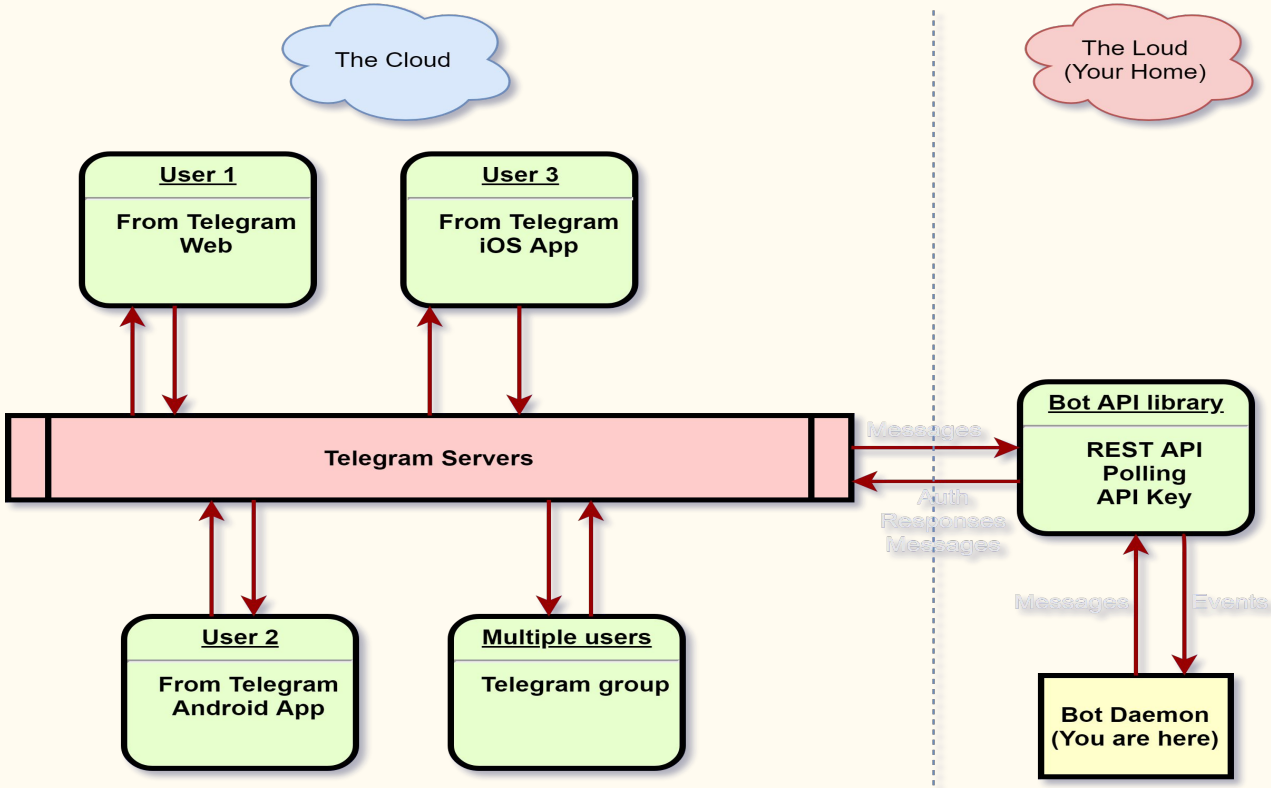
Manu Sánchez & Javier G. Sogo

Objectives

- Spend the evening **together**
- Get an introduction to a **cool library**
- Run a **working example** on your computer
- Help your colleagues to **understand** how the example works
- Try, experiment... **learn**
- **Share** your achievements

Theory

Arquitectura básica de un bot



Motivación: Casos de uso

- (Do It Yourself) Smart Home
- Wake on LAN setup
- Newsletter
- Bricklink price drops notifications
- Wallapop!

Why telegram?

- Chat oriented: Easy to interact with
- Easy setup: Get API key, launch your program, done
- Powerful: Not only messaging but commands, pictures, games, etc
- OSS library implementations of the Bot API

tgbot-cpp: C++ library for Telegram bot API

- https://github.com/reo7sp/tgbot-cpp
- Object oriented abstraction of the Bot API: `TgBot::Bot`, `TgBot::Message`, `TgBot::Chat`, etc
- Simple listening API through event handlers (callbacks)
- Different event handlers for incoming commands or messages

The code

Core concepts: echo bot

Create the bot

```
TgBot::Bot bot("PLACE YOUR TOKEN HERE");
```

Connect functions to commands

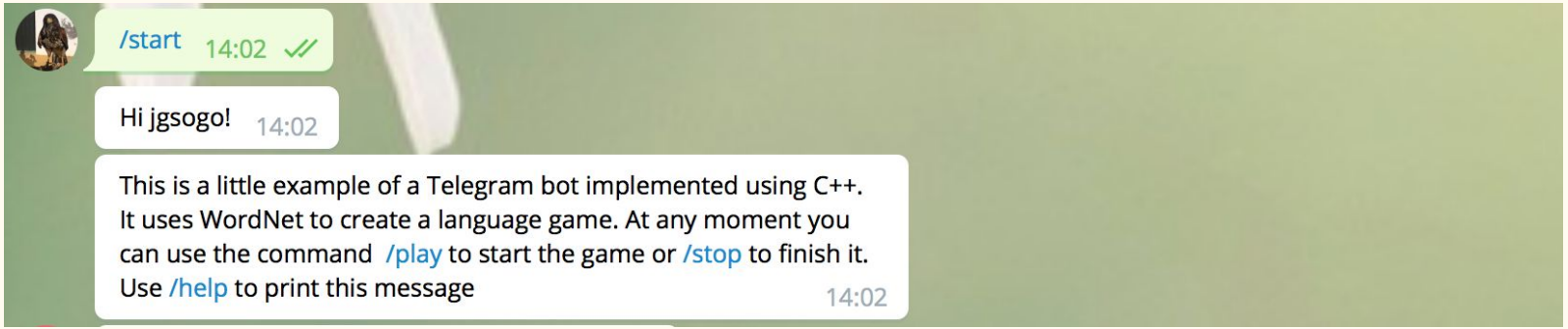
```
bot.getEvents().onCommand("start", [&bot](TgBot::Message::Ptr message) {
    bot.getApi().sendMessage(message->chat->id, "Hi!");
});
bot.getEvents().onAnyMessage([&bot](TgBot::Message::Ptr message) {
    printf("User wrote %s\n", message->text.c_str());
    if (StringTools::startsWith(message->text, "/start")) {
        return;
    }
    bot.getApi().sendMessage(message->chat->id, "Your message is: " + message->text);
});
```

Core concepts

Run polling

```
TgBot::TgLongPoll longPoll(bot);
while (true) {
    printf("Long poll started\n");
    longPoll.start();
}
```

Talk!



Ej1: Chatroulette

A bot that connects two users randomly and let them interchange messages

```
void BotRandom::initialize() {
    _bot.getEvents().onCommand("start", [this](TgBot::Message::Ptr message){ this->start(message);});
    _bot.getEvents().onCommand("stop", [this](TgBot::Message::Ptr message){ this->stop(message, true);});
    _bot.getEvents().onCommand("help", [this](TgBot::Message::Ptr message){ this->help(message);});
    _bot.getEvents().onCommand("report", [this](TgBot::Message::Ptr message){ this->report(message);});

    _bot.getEvents().onAnyMessage([this](TgBot::Message::Ptr message){ this->on_message(message);});
}
```

https://github.com/madridccppug/workshop-telegram-bot/tree/master/random_chat

Ej2: WordNet Trivial

A bot that ask for the english word given a definition

```
void Bot::initialize() {
    _bot.getEvents().onCommand("start", [this](TgBot::Message::Ptr message){ this->start(message);});
    _bot.getEvents().onCommand("stop", [this](TgBot::Message::Ptr message){ this->stop(message);});
    _bot.getEvents().onCommand("play", [this](TgBot::Message::Ptr message){ this->play(message);});
    _bot.getEvents().onCommand("help", [this](TgBot::Message::Ptr message){ this->help(message);});

    _bot.getEvents().onAnyMessage([this](TgBot::Message::Ptr message){ this->on_message(message);});
}
```

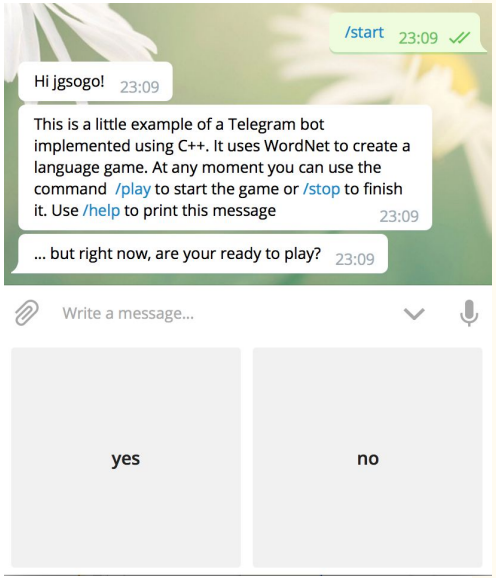
https://github.com/madridccppug/workshop-telegram-bot/tree/master/wordnet_game

https://wordnet.princeton.edu/

Ej2: WordNet Trivial

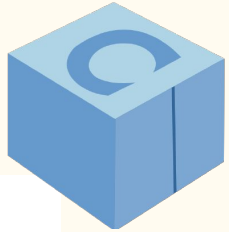
You can create a keyboard with predefined options!

```
TgBot::ReplyKeyboardMarkup::Ptr yes_no_keyb() {
    TgBot::ReplyKeyboardMarkup::Ptr keyboard(new TgBot::ReplyKeyboardMarkup);
    keyboard->oneTimeKeyboard = true;

    std::vector<TgBot::KeyboardButton::Ptr> row;
    TgBot::KeyboardButton::Ptr yes(new TgBot::KeyboardButton);
    yes->text = "yes";
    row.push_back(yes);
    TgBot::KeyboardButton::Ptr no(new TgBot::KeyboardButton);
    no->text = "no";
    row.push_back(no);

    keyboard->keyboard.push_back(row);
    return keyboard;
}
```

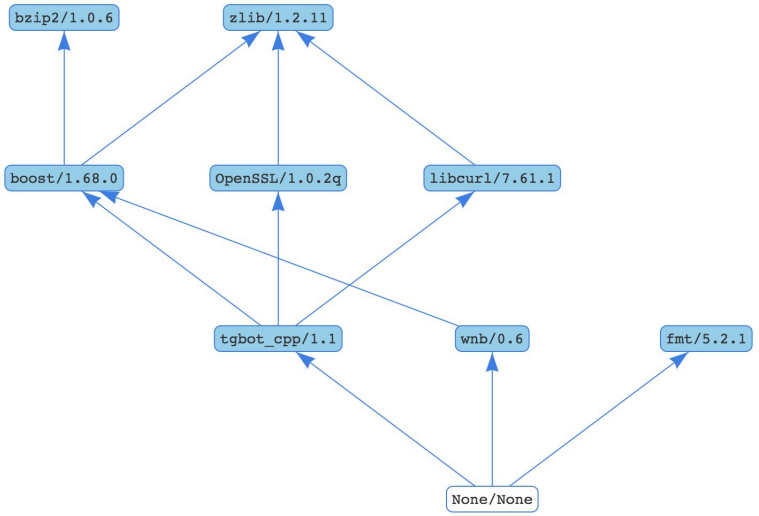


```
bool inserted; std::map<int32_t, _t_actions>::iterator it;
std::tie(it, inserted) = _callbacks.insert(std::make_pair(message->chat->id, _t_actions()));

for (auto& option: q) {
    if (option.first == chosen) {
        it->second[option.first] = [this](TgBot::Message::Ptr msg) {
            std::stringstream ss; ss << "Great! +1 point! " << emoji::green_check;
            _bot.getApi().sendMessage(msg->chat->id, ss.str());
            this->play(msg);
        };
    }
    else {
        it->second[option.first] = [this, q, chosen](TgBot::Message::Ptr msg) {
            std::stringstream ss; ss << emoji::red_cross << " Nooo... *" << msg->text << "* means " << emoji::book << " _"
                ss << q.at(msg->text) << "_. We were looking for *" << chosen << "*.";
            _bot.getApi().sendMessage(msg->chat->id, ss.str(), false, 0, std::make_shared< TgBot::GenericReply >(), "Markd
            this->play(msg);
        };
    }
    std::vector<TgBot::KeyboardButton::Ptr> row;
    TgBot::KeyboardButton::Ptr button(new TgBot::KeyboardButton);
    button->text = option.first;
    row.push_back(button);
    keyboard->keyboard.push_back(row);
}
```

There are a couple of dependencies...



```
class TelegramWordnet(ConanFile):
    settings = "os", "compiler", "build_type", "arch"
    generators = "cmake"

    def requirements(self):
        self.requires("tgbot_cpp/1.1@jgsogo/stable")
        self.requires("wnb/0.6@jgsogo/stable")
        self.requires("fmt/5.2.1@bincrafters/stable")

    def build(self):
        cmake = CMake(self)
        cmake.configure()
        cmake.build()

    def imports(self):
        self.copy("*.dll", dst="bin", src="bin")
        self.copy("*.dylib*", dst="bin", src="lib")
        self.copy('*.so*', dst='bin', src='lib')

        self.copy("*", dst="bin/wordnet", src='data', keep_path=True)
```



Hands-on

Get and compile the sources

Conan related stuff

```
$> pip install conan

$> conan remote add bincrafters https://api.bintray.com/conan/bincrafters/public-conan
$> conan remote add jgsogo https://api.bintray.com/conan/jgsogo/conan-packages
```

Clone the repo

```
$> git clone https://github.com/madridccppug/workshop-telegram-bot
$> git submodule update --init --recursive
```

Compile

```
$> cd <path/to/example>
$> mkdir build && cd build
$> conan install .. --build=missing
$> cmake .. -DCMAKE_BUILD_TYPE=Release
$> cmake --build .
```

Talk to BotFather

1) Go to https://telegram.me/BotFather
2) `/help`
3) Create your BOT
4) Get your TOKEN

Build your own, or modify ours

Proof on concept

- Echo bot
- Simple math bot

Challenges:

- Grab a definition from RAE
- A bot to notify new meetups
- Bricklink price drops

Share