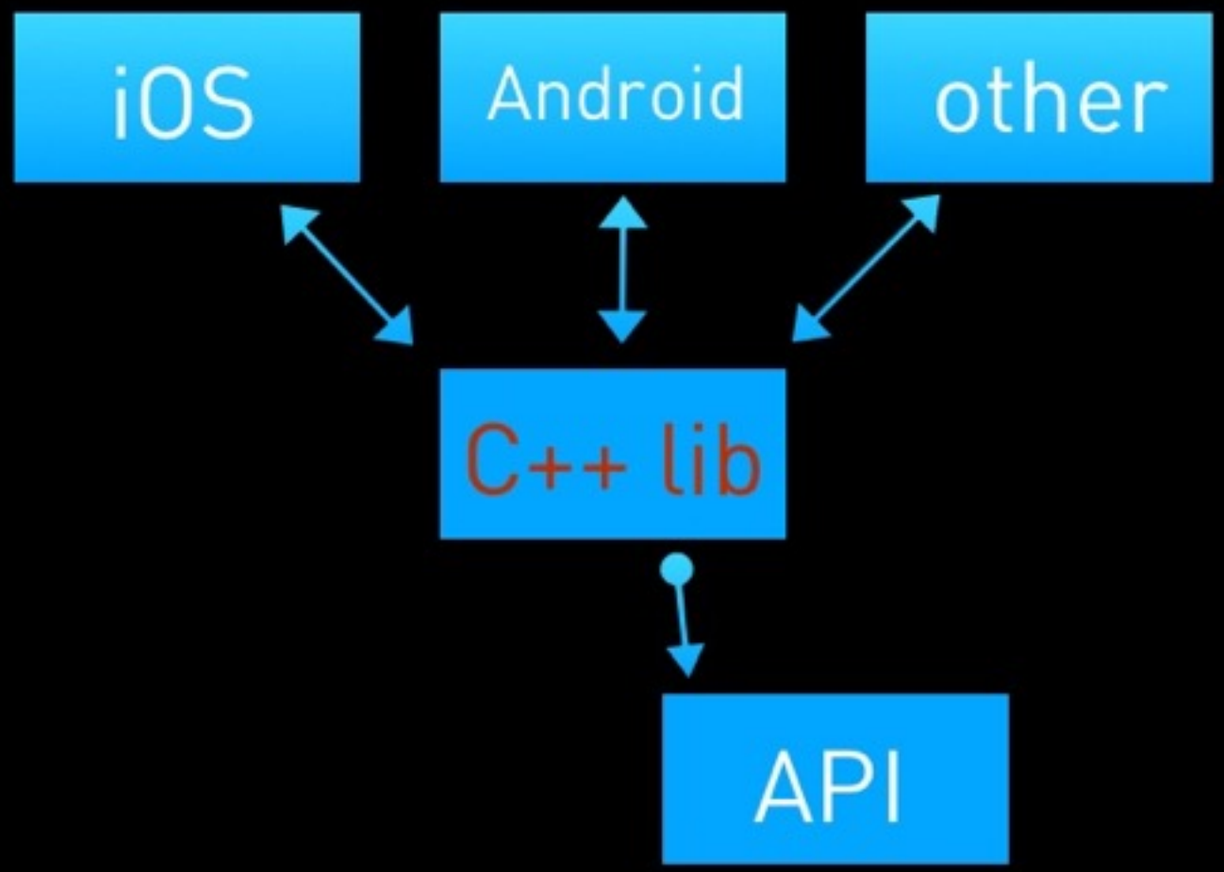
2x more effective development ingeniously bundling your app's business logic

Adas Burkšaits ir Tadas Razmislavičius

Apžvalga

- Pristatysim C++ shared library
- Naudojimas iš iOS pusės
- Pavyzdžiai
- Ateities planai

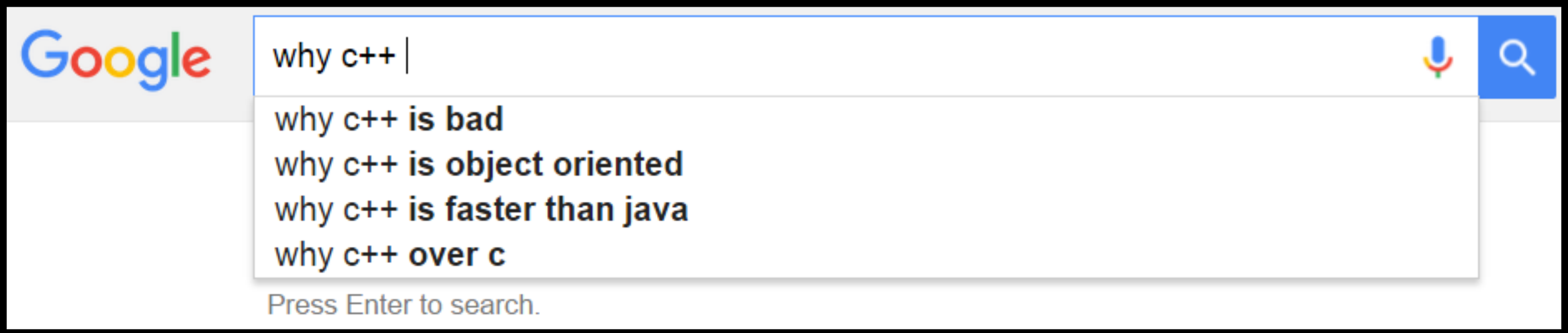
Architektūra



Motyvacija

- Sudėtinga logika kliente
- Dukart mažiau kodo
- Sunku surasti mobilių platformų programuotojus

C++


Google
why c++
why c++ is bad
why c++ is object oriented
why c++ is faster than java
why c++ over c
Press Enter to search.

C++ cons

- Atminties valdymas
- Federation of languages
- Nerangumas
- Trečios šalies bibliotekų integracija
- Sunku pradėti

C++ pros

- C++11 & C++14
- Greitis
- Assembly
- Galima daryti viską
- Yra geriau nei atrodo!

Bridging pain

```
CJNIEXPORT jint JNICALL Java_com_trl_TimesTabVm_00024CppProxy_native_1getCellCount(JNIEnv* jniEnv, jobject /*this*/, jlong nativeRef)
{
    try {
        DJINNI_FUNCTION_PROLOGUE1(jniEnv, nativeRef);
        const auto& ref = ::djinni::objectFromHandleAddress<::trl_gen::TimesTabVm>(nativeRef);
        auto r = ref->get_cell_count();
        return ::djinni::release(::djinni::I32::fromCpp(jniEnv, r));
    } JNI_TRANSLATE_EXCEPTIONS_RETURN(jniEnv, 0 /* value doesn't matter */)
}


CJNIEXPORT jobject JNICALL Java_com_trl_TimesTabVm_00024CppProxy_native_1getCellAtIndex(JNIEnv* jniEnv, jobject /*this*/, jlong nativeRef, jint j_cellIndex)
{
    try {
        DJINNI_FUNCTION_PROLOGUE1(jniEnv, nativeRef);
        const auto& ref = ::djinni::objectFromHandleAddress<::trl_gen::TimesTabVm>(nativeRef);
        auto r = ref->get_cell_at_index(::djinni::I32::toCpp(jniEnv, j_cellIndex));
        return ::djinni::release(::djinni_generated::NativeHourVm::fromCpp(jniEnv, r));
    } JNI_TRANSLATE_EXCEPTIONS_RETURN(jniEnv, 0 /* value doesn't matter */)
}
```

```
CJNIEXPORT jobject JNICALL Java_com_trl_TimesTabVm_00024CppProxy_native_1getHours(JNIEnv* jniEnv, jobject /*this*/, jlong nativeRef)
{
    try {
        DJINNI_FUNCTION_PROLOGUE1(jniEnv, nativeRef);
        const auto& ref = ::djinni::objectFromHandleAddress<::trl_gen::TimesTabVm>(nativeRef);
        auto r = ref->get_hours();
        return ::djinni::release(::djinni::List<::djinni_generated::NativeHourVm>::fromCpp(jniEnv, r));
    } JNI_TRANSLATE_EXCEPTIONS_RETURN(jniEnv, 0 /* value doesn't matter */)
}

CJNIEXPORT jobject JNICALL Java_com_trl_TimesTabVm_00024CppProxy_native_1getCommentsMarkers(JNIEnv* jniEnv, jobject /*this*/, jlong nativeRef)
{
    try {
        DJINNI_FUNCTION_PROLOGUE1(jniEnv, nativeRef);
        const auto& ref = ::djinni::objectFromHandleAddress<::trl_gen::TimesTabVm>(nativeRef);
        auto r = ref->get_comments_markers();
        return ::djinni::release(::djinni::List<::djinni_generated::NativeTimesCommentVm>::fromCpp(jniEnv, r));
    } JNI_TRANSLATE_EXCEPTIONS_RETURN(jniEnv, 0 /* value doesn't matter */)
}

CJNIEXPORT jobject JNICALL Java_com_trl_TimesTabVm_00024CppProxy_native_1onClick(JNIEnv* jniEnv, jobject /*this*/, jlong nativeRef, jint j_indexHour, jint j_indexMinute)
{
    try {
        DJINNI_FUNCTION_PROLOGUE1(jniEnv, nativeRef);
        const auto& ref = ::djinni::objectFromHandleAddress<::trl_gen::TimesTabVm>(nativeRef);
        auto r = ref->on_click(::djinni::I32::toCpp(jniEnv, j_indexHour),
                               ::djinni::I32::toCpp(jniEnv, j_indexMinute));
        return ::djinni::release(::djinni_generated::NativeRunVm::fromCpp(jniEnv, r));
    } JNI_TRANSLATE_EXCEPTIONS_RETURN(jniEnv, 0 /* value doesn't matter */)
}
```

Djinni!

- Dropbox atviro kodo projektas
- Generuoja C++, Java ir Obj-C iš djinni interface definition failo
- Generuoja kodą, kuris leidžia kviesti C++ iš ObjC ir atvirkščiai
- Tipai: *Enum*, *Record*, *Interface*

```
my_record = record {
    id: i32;
    info: string;
    store: set<string>;
    hash: map<string, i32>;

    values: list<another_record>;

    # Comments can also be put here

    # Constants can be included
    const string_const: string = "Constants can be put here";
    const min_value: another_record = {
        key1 = 0,
        key2 = ""
    };
}

another_record = record {
    key1: i32;
    key2: string;
} deriving (eq, ord)

# This interface will be implemented in C++ and can be called from any language.
my_cpp_interface = interface +c {
    method_returning_nothing(value: i32);
    method_returning_some_type(key: string): another_record;
    static get_version(): i32;

    # Interfaces can also have constants
    const version: i32 = 1;
}

# This interface will be implemented in Java and ObjC and can be called from C++.
my_client_interface = interface +j +o {
    log_string(str: string): bool;
}
```

Project config

- C++ projektą ir jo bibliotekas reikia pridėti prie jūsų mobile app
- *xcodeproj*
- *Android.mk*

GYP

- **G**enerate **Y**our **P**rojects!
- Input: json projekto aprašymas
- Output: xcodeproj, Android.mk, …

GYP

```
{
    'targets': [
                {
                'target_name': 'new_unit_tests',
                'type': 'executable',
                'defines': [
                            'FOO',
                            ],
                'include_dirs': [
                             '..',
                             ],
                'dependencies': [x
                             'other_target_in_this_file',
                             'other_gyp2:target_in_other_gyp2',
                             ],
                'sources': [
                            'new_additional_source.cc',
                            'new_unit_tests.cc',
                            ],
                },
                ],
}
```

Bibliotekų integravimas

- Bibliotekos dažniausiai turi ./configure
- Skirtingos mobiliųjų architektūros, taigi reikia konfigūruoti pačiam
- C++ low-level architecture specific optimizations

Naudojimas iš iOS pusės

- Tik UI implementacija
- Data modelis jau paruoštas naudojimui
- Patogus interfeisas (nullability, strongly typed data sets)

Djinni IDL

```
@import "http.djinni"
@import "events.djinni"
@import "schedule_response.djinni"
@import "get_stops_and_tracks_response.djinni"
@import "offline_status.djinni"
@import "vm_callback.djinni"

api = interface +c {
    static create_api_with_url(api_url: string,
        http_impl: http): api;
    static get_api(): api;
    get_schedule(scheduleId: string,
        callback: api_schedule_callback);
    get_stops_and_tracks(userLocationId: string,
        q: string,
        lat: f64,
        lng: f64,
        callback: api_stops_and_tracks_callback);
    get_offline_status(user_location_id: string): offline_status;
    save_offline(data: binary,
        callback: event);
}
```

C++

```cpp
// AUTOGENERATED FILE - DO NOT MODIFY!
// This file generated by Djinni from trl.djinni

#pragma once

#include "offline_status.hpp"
#include <cstdint>
#include <memory>
#include <string>
#include <vector>

namespace trl_gen {

class ApiScheduleCallback;
class ApiStopsAndTracksCallback;
class Event;
class Http;

class Api {
public:
    virtual ~Api() {}

    static std::shared_ptr<Api> create_api_with_url(const std::string & api_url,
                                                    const std::shared_ptr<Http> & http_impl);

    static std::shared_ptr<Api> get_api();

    virtual void get_schedule(const std::string & scheduleId,
                              const std::shared_ptr<ApiScheduleCallback> & callback) = 0;

    virtual void get_stops_and_tracks(const std::string & userLocationId,
                                      const std::string & q,
                                      double lat,
                                      double lng,
                                      const std::shared_ptr<ApiStopsAndTracksCallback> & callback) = 0;

    virtual OfflineStatus get_offline_status(const std::string & user_location_id) = 0;

    virtual void save_offline(const std::vector<uint8_t> & data,
                              const std::shared_ptr<Event> & callback) = 0;
};

}  // namespace trl_gen
```

Objective-C

```objc
// AUTOGENERATED FILE - DO NOT MODIFY!
// This file generated by Djinni from trl.djinni

#import "TRLFavoriteScheduleDepartureIdDto.h"
#import "TRLOfflineStatus.h"
#import <Foundation/Foundation.h>
@class TRLApi;
@protocol TRLApiScheduleCallback;
@protocol TRLApiStopsAndTracksCallback;
@protocol TRLEvent;
@protocol TRLHttp;


@interface TRLApi : NSObject

+ (nullable TRLApi *)createApiWithUrl:(nonnull NSString *)apiUrl
                             httpImpl:(nullable id<TRLHttp>)httpImpl;

+ (nullable TRLApi *)getApi;

- (void)getSchedule:(nonnull NSString *)scheduleId
           callback:(nullable id<TRLApiScheduleCallback>)callback;

- (void)getStopsAndTracks:(nonnull NSString *)userLocationId
                        q:(nonnull NSString *)q
                      lat:(double)lat
                      lng:(double)lng
                 callback:(nullable id<TRLApiStopsAndTracksCallback>)callback;

- (nonnull TRLOfflineStatus *)getOfflineStatus:(nonnull NSString *)userLocationId;

- (void)saveOffline:(nonnull NSData *)data
           callback:(nullable id<TRLEvent>)callback;

@end
```

Djinni IDL

```
@import "threading.djinni"

client_api = interface +o +j {
    get_home_dir(): string;
    get_current_location(): optional<lat_lng>;
    get_locatization(key: string): string;
    get_svg_image(name:string, size:i32, color:optional<string>) : optional<binary>;
    fatal_error(error_msg:string);
    execute(task: async_task);
}

api = interface +c {
    static initialise(platform_config: platform_config);
    ...
}
```

Screen view model

```objc
// AUTOGENERATED FILE - DO NOT MODIFY!
// This file generated by Djinni from run_vm.djinni

#import "TRLRunStopCellVm.h"
#import "TRLVmStatus.h"
#import <Foundation/Foundation.h>
@class TRLRunVm;
@protocol TRLEvent;


@interface TRLRunVm : NSObject

// view model default methods:

+ (nullable TRLRunVm *)create:(nonnull NSString *)scheduleId
                       stopId:(nonnull NSString *)stopId
                        runId:(nonnull NSString *)runId
                realtimeRunId:(nonnull NSString *)realtimeRunId;
- (void)subscribe:(nullable id<TRLEvent>)e;
- (void)unsubscribe;
- (void)load;
- (TRLVmStatus)getStatus;
- (nonnull NSString *)getMessage;
// view model specific methods:
- (nonnull NSString *)getTitle;
- (int32_t)getCellsCount;
- (nullable TRLRunStopCellVm *)getStopAtIndex:(int32_t)index;
- (nonnull NSString *)getScheduleId;
- (nonnull NSString *)getStopId;
- (nonnull NSString *)getRunId;
- (nonnull NSString *)getRealtimeRunId;
@end
```

```swift
//
//  RunViewController.swift
//  Trafi
//
//  Created by Tadas Razmislavicius on 25/08/15.
//  Copyright (c) 2015 Intelligent Communications. All rights reserved.
//

import Foundation

class RunCell : UITableViewCell {
    @IBOutlet weak var timeLabel: UILabel!
    @IBOutlet weak var nameLabel: UILabel!
}


class RunViewController: UITableViewController, TRLEvent {

    var viewModel:TRLRunVm! {
        willSet {
            self.viewModel?.unsubscribe()
        }
        didSet {
            self.viewModel?.subscribe(self)
            self.viewModel?.load()
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    func updated() {
        dispatch_async(dispatch_get_main_queue(), {
            self.navigationItem.title = self.viewModel?.getTitle() ?? ""
            self.tableView?.reloadData()
        })
    }

    override func numberOfSectionsInTableView(tableView: UITableView) -> Int {
        return 1
    }

    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return self.viewModel?.getCellsCount().int ?? 0
    }

    override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
        let cellVM = viewModel.getStopAtIndex(indexPath.row.i32)!
        let cell = tableView.dequeueReusableCellWithIdentifier("RunCell") as! RunCell

        cell.timeLabel.text = cellVM.timeText ?? ""
        cell.nameLabel.text = cellVM.name ?? ""

        return cell
    }
}
```

Platformos implementacija

- Http
- Multithreading

Dabartinėje versijoje

- 10 000 C++ kodo eilučių
- Ikonų paišymas
- Dalis WEB API metodų, data modelis
- Offline tvarkaraščiai

Planai

- Pabaigti migraciją į C++
- Offline maršruto paieška