



# Programming Mobile Applications in Flutter

Introduction and Dart

# About us

- mgr inż. Rafał Adasiewicz
- Contact: [adasiewicz.r@gmail.com](mailto:adasiewicz.r@gmail.com)
- Android Team Lead at Stream
- > 6 years of mobile applications development
- Android, iOS, Flutter, React Native

# About us

- inż. Mateusz Wojtczak
- Contact: [mateusz@leancode.pl](mailto:mateusz@leancode.pl)
- Mobile Guild Leader at LeanCode
- > 5 years of mobile applications development
- Flutter, React Native, iOS, Xamarin, Android



# Who are you?

# Introduction

- Programming Mobile Application in Flutter
  - Lectures
  - Labs
  - Project
- Materials - <https://github.com/leancodepl/flutter-at-mini>
- Consultations
  - Online



+

+

+

Dołącz do nas!



Flutter Developer

...

...ale też React i .Net Core Devs również mile widziani ;)

+

+

+



# What is it all about?



# Rules

- Points 0-100:
  - **51-60pt - 3**
  - **61-70pt - 3.5**
  - **71-80pt - 4**
  - **81-90pt - 4.5**
  - **91-100pt - 5**
- Project - 100pt
- Activity during lectures - 10pt

# Lectures

1. Introduction and Dart
2. Introduction to Flutter 1
3. Introduction to Flutter 2
4. State Management
5. Async and HTTP
6. State Management with External Libraries
7. Architecture and Dependency Injection
8. Storing Data

# Lectures

- Forms
- Firebase
- Code Generation and Internationalization
- Animations
- Flutter Web and Flutter Desktop
- Communication with Native
- **Waiting for Proposals**

# Labs

1. Getting Started with Flutter
2. Layouts 1
3. Layouts 2
4. Communication with API
5. State Management
6. CodeMagic
7. Authorization

# Project

- Individual multi-layer Flutter application that works at least on one mobile platform (Android/iOS)
- Application should contain at least two screens
- Application should communicate with 3rd party API OR use other data persistency solution
- Application's topic and scope is defined by the student, should be described in initial documentation and approved by the lecturer

# Sample Projects

- TODO List with authorization and synchronization between devices
- Chat with authorization
- Shopping list with categories, search, history
- Feed using 3rd party API
- Online shop with deep links, categories, filters, sort, cart
- Pol browser - map and list, tags, categories, sort by localization

# Documentation

- Initial Documentation
  - Project Description
  - Use cases
- Final Documentation
  - Project description
  - Integrations
  - List of optional requirements
  - Instruction
  - Test account (if applicable)
  - Database/Firestore schema (if applicable)
  - CI/CD description/screenshot (if applicable)

# Sample initial Documentation

- Chat with Authorization
- Description
  - Screens list and short description
    - Login Screen
    - Channel List Screen
    - Message List Screen
- Use cases
  - As a User, I can sign in using Google/Facebook/Instagram/Apple account
  - As a User, I can see a list of channels
  - As a User, I can create/delete/leave channel if I have sufficient permissions
  - As a User, I can send plain text messages
  - As a User, I can send images/videos/files
  - As a User, I can edit messages
  - As a User, I can reply in thread



# Assessment Rules

- Implementation of the required project assumptions **50pt**
  - Initial documentation - **5pt**
  - Implementation of a multi-layer application - **15pt**
  - Code quality - **10pt**
  - UI/UX - **10pt**
  - Final documentation - **10pt**
- **Adherence to the schedule - 10pt**

# Assessment Rules

- Optional requirements (**max 50pt**)
  - Support for additional platform (Android/iOS/Web/Desktop) - **5pt** each
  - Implementing BLoC pattern - **10pt**
  - Animations - **10pt**
  - Tests - **10pt**
  - Signing in process - **10pt**
  - Complex form with validation - **10pt**
  - CI/CD - **5pt**
  - Platform Channels - **10pt**
  - Internationalization - **5pt**
  - Accessibility - **5pt**
  - Custom painting - **10pt**
  - Deep links - **10pt**
  - Using Camera/Bluetooth/Other platform features - **10pt**
  - Offline support - **20pt**

# Timeline

- **29.10.2021** - Initial documentation (Labs)
- **28.01.2022** - Project Submission
  - Source Code and Final Documentation
- **11.02.2022** - Late Project Submission
  - Each day of being late will take a decrease of **5pt** from the total number of gained points

# Resources

- Flutter Official Documentation - <https://flutter.dev/docs>
- Code Documentation
- Pub Dev - <https://pub.dev>
- Alberto Miola - Flutter Complete Reference

# QUESTIONS?

# Dart

# Dart

- Object-oriented, class-based, garbage-collected, C-style syntax programming language
- Open source, developed by Google
- First stable release - 2011
  - Designed for Web
  - Dart VM in Chrome
  - Optional type system
- Dart 2.0 - August 2018
  - Static type system
  - *dynamic*
- Dart 2.12.0 - March 2021
  - Null-safety

# Dart

- Designed for client development
  - Optimized for UI
  - Productive Development - Make changes iteratively: use hot reload to see the result instantly in your running app
- Compile to ARM & x64 machine code for mobile, and desktop. Or compile to JavaScript for the web
- Dart VM with just-in-time (JIT) compilation and an ahead-of-time (AOT) compiler for producing machine code.



# Why Dart?

# Why Dart?

- Flutter used four primary dimensions for evaluation, and considered the needs of framework authors, developers, and end users:
  - Developer productivity
  - Object-orientation
  - Predictable, high performance
  - Fast allocation
- **Opportunity to work closely with the Dart community, which is actively investing resources in improving Dart for use in Flutter**

# DART CHEAT SHEET PDF

A Dart Language PDF for C# and Java Developers

## PRIMITIVES

bool, int, double  
String, List, Map, Set

## FUNCTIONS

return-type name (parameters) {body}  
return-type name (parameters) => expression;

## OPTIONAL PARAMETERS

void foo(string arg1, [int arg2 = 0, int arg3 = 0]) {...} Positional  
void foo(string arg1, {int arg2 = 0, int arg3 = 0}) {...} Named

## PROPERTIES

string get ClientName => \_clientName;  
string set ClientName(string s) { \_clientName = s; }

## CONSTRUCTORS

Point(double x, double y) {...}	Default
Point.asPolar(double angle, double r) {...}	Named
Client(String this_name) {...}	this instance initializer
Customer(String name) : _code = _name {...}	Member initializer

## MODIFIERS

int _x	Private because of underscore
var a = 1;	Variable
final b = a + 1;	Runtime constant
const c = 3;	Compile-time constant

## A FEW OPERATORS

cmp ..name = 'Alice' ..supervisor = 'Zolttron' ..hire();	Cascade
var smallList = [1, 2]; var bigList = [0, ...smallList, 3, 4];	Spread
if (obj is String) ...	Type test
if (obj is! String) ...	Negative type test
print(message ?? "none");	Null-coalesce (use right expression if left is null)
x ??= 1;	Assign only if x was null
client?.name;	Null-aware (returns null if client is null)
count ~/ 100;	Integer division

Source:

<https://www.toptal.com/dart/dartlang-guide-for-csharp-java-devs>

# DartPad



# Show me the code!



# Questions?