LeanCode

Programming Mobile Applications in Flutter

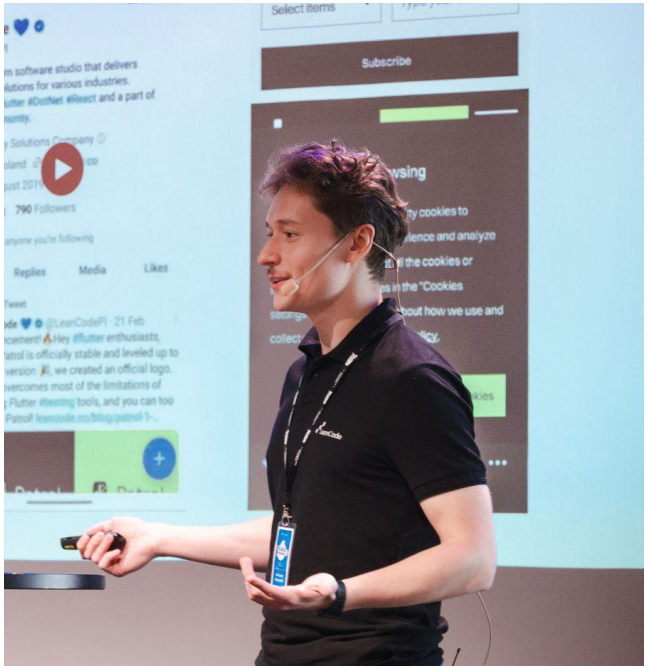
Intro lecture

Who are we?

- mgr inż. Jakub Fijałkowski
- Contact: jakub.fijalkowski@leancode.pl
- Backend Guild Leader @ LeanCode
- ~10 yrs of backend development, some experience with mobile
- Rust, .NET, Cloud, DevOps
- Books, Golf, Gigs

Who are we?

- inż. Mateusz Wojtczak
- Contact: mateusz.wojtczak@leancode.pl
- Head of Mobile @ LeanCode
- Flutter, React Native, Xamarin, Native
- Google Developer Expert
- producing music after work



Who are we?

- inż. Piotr Rogulski
- Contact: piotr.rogulski@leancode.pl
- Flutter Developer @ LeanCode
- Fluttercon Berlin 2024 speaker
- Author of bloc_lens package

LeanCode

As LeanCode, we sponsored all major **Flutter events**:

- ❑ FlutterCon Berlin,
- ❑ Flutter & Friends Stockholm,
- ❑ Flutter Firebase Festival Prague.

LeanCode

Who are you?

What is it all about?

https://github.com/leancodepl/flutter-at-mini

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
- Labs (non-obligatory) - 8 * 3 = 24 pt

Lectures

1. Intro
2. What is Flutter?
3. Let’s go deeper into Flutter
4. State Management
5. Async and HTTP
6. State Management with External Libraries
7. Firebase
8. Data persistence

Lectures

9. Testing
10. Forms
11. Architecture
12. Flutter Web
13. Platform integration
14. Animations - guest lecture
15. TBA (GameDev?)

Labs

1. Getting Started with Flutter
2. Layouts 1
3. Layouts 2
4. State Management
5. Communication with API
6. State Management with External Libs
7. Firebase & Authentication
8. **It depends on you :)**

Weeks 9-15 → project consulting

Project

- Individual multi-layer Flutter application that works at least on one mobile platform (Android/iOS)
- Project topic and scope is defined by the student, should be described in the initial documentation and approved by the lecturer

Sample projects from past editions

- Collaborative live whiteboard
- Shipment & delivery app
- “Ships” game
- “4 in a row” game
- Resume PDF generator app

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)

Example initial docs

- 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

Assessment Rules

Implementation of the required project assumptions (50pt):

- Initial documentation - 5pt
- Architecture - 15pt
- Code quality (e.g. static code analysis, formatting) - 15pt
- UI/UX
  - Material Design - 5pt
  - Custom design widgets - 5pt
- Final documentation - 5pt

Assessment Rules

Optional requirements (**max 50pt)**

- Support for each additional platform (Mobile/Web/Desktop) - 5pt each
- Animations
  - Implicit / ready-to-use packages - max 5pt
  - Custom - max 10pt
- Tests
  - Unit tests - max 5pt
  - Widget tests - max 5pt
  - Patrol tests - max 10pt
- Signing in process
  - Firebase Auth - max 5pt
  - Custom-backend auth - max 10pt
- Multi-step form with validation - max 10pt

Assessment Rules

Optional requirements (**max 50pt)**

- CI/CD - max 15pt
  - code analysis & run flutter test - max 10pt
  - deploy app - max 10pt
- Platform Channels
  - Using pub package for platform features (e.g. camera) - 5pt
  - Creating custom platform channels - 15pt
- Internationalization - max 10pt
- Custom painting - max 5pt
- Local data persistence (offline) - max 15pt

Timeline

- **08.11.2024** - Initial documentation (Lab 4)
- **28.01.2025** - Project Submission
  - Source Code and Final Documentation
- **12.02.2024** - Late Project Submission
  - Each day of being late will take a decrease of **5pt** from the total number of gained points

Any questions?

Dart

Dart

- Statically typed
- Object-oriented
- Garbage-collected
- ~~C-style syntax~~ modern-ish syntax
- Asynchrony support
- 100% sound null safe
- Open source, developed by Google

```dart
int fibonacci(int n) {
  if (n == 0 || n == 1) return n;
  return fibonacci(n - 1) + fibonacci(n - 2);
}

var result = fibonacci(20);
```

```dart
var (a, b) = ('left', 'right');
(b, a) = (a, b); // Swap.
print('$a $b'); // Prints "right left".
```

Dart

- Designed for client development
  - Optimized for UI & Flutter
  - Productive Development - Make changes iteratively: use hot reload to see the result instantly in your running app
- Compiled to ARM & x64 machine code for mobile, and desktop.
- Compiled to **WebAssembly** or JavaScript on 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**

Why Dart? - TLDR

- It looks like JavaScript
- It is fast enough (we have 16.6 ms to render a frame)
- It’s garbage-collected
- It can perform tree shaking to reduce code size
- It’s tightly connected to Flutter use cases
- It’s targeted to run on mobile/desktop/web.

dartpad.dev

idx.dev

Let’s jump into code

Questions?