- title : Design patterns?
- description : Design patterns, but no
- author : Krystian Kolad
- theme : night
- transition : default

~~Design patterns~~

About working with Code

Who am I?

Agenda

- Conventions
- A word about architecture
- Design patterns
- SOLID
- What else?
- Q&A

Conventions

- Variables naming
- Folders
- Classes and Methods naming

A word about architecture

- What is it?
- Whay should I care?

Why do we need design patterns?

- Helps us organize our code
- Provides a proven solutions

Good design patterns

Repository pattern

- Abstraction between database and service
- Easy to use
- Helpfull on every stage of work

Factory

- Provider for instances of other classes
- EAsy to use

Bad design patterns

GOD object

- One class does everything
- Hard to maintain

Singleton

- One instance of object in whole program
- Can cause unwanted troubles, e.g. deadlocks

SOLID principles

Single responsibility principle

Class should have one and only one responsibility

Open/closed principle

Class should be opened to enlargement, but closed to modification

---

Liskov substitution principle

Every children of a class should be able to replace it parent without any cause

---

Interface segregation principle

Interfaces should be specialized

Dependency inversion principle

Classes should be dependend on abstraction, no concrete implementation

---

What else?

KISS

Keep it Sipmle, Stupid

- More complex your code is, harder it will be to maintain it
- Do not complicate anything!

DRY

Don't Repeat Youtself

- If you use some login in more than one place, move it to some service
- It's better to maintain one service than 3 same pieces of code whic you have to update

---

YAGNI

You Aren't Gonna Need It

- Do not write anything, because "maybe you will need it". You won't
- Even if you will, I bet it will be outdated then

Q & A

Contact Me

- https://www.facebook.com/KrystianKolad
- https://github.com/KrystianKolad
- https://piastdotnet.github.io/