Jak być (jeszcze) lepszym programistą

Bartosz Gołek

O mnie

Programista – 10 lat doświadczenia

Technologie: PHP, C#, Python

Zainteresowania: OOP, TDD, BDD

Obowiązki:

Projektowanie i tworzenie rozwiązań dla systemów rozproszonych

Agenda

- **Wstęp**
- **Zarys pojęć**
- **O architekturze**
- **Trening czyni mistrza**
- **Pytania**

Co to znaczy być dobrym programistą?

- kompletność kodu?
- czytelność kodu?
- znajomość algorytmów (wydajność)?
- kod podatny na zmiany?
- ilość błędów?
- szybkość pisania kodu?

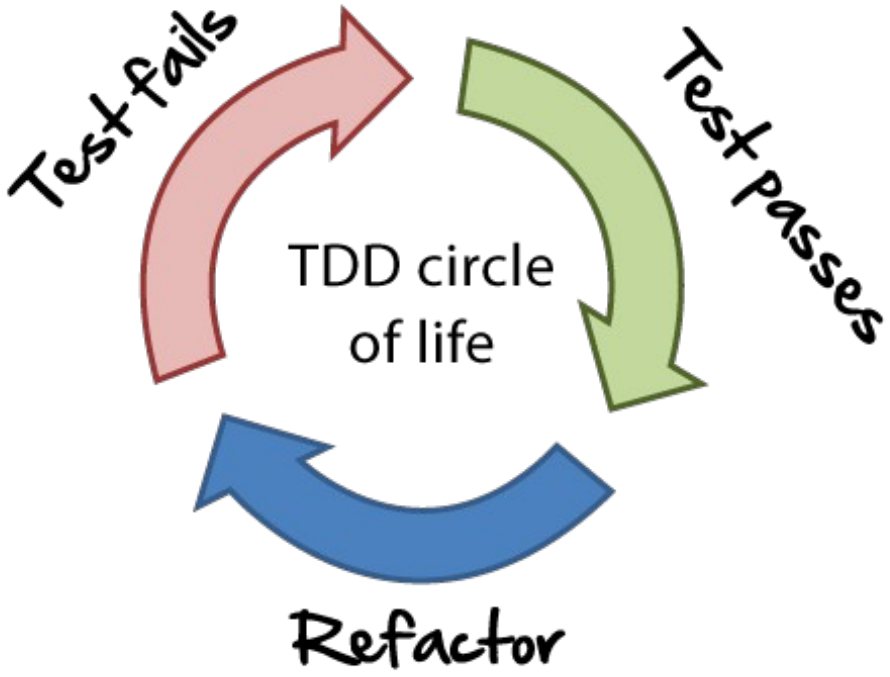
SOLID

- **S**ingle Responsibility Principle
- **O**pen Closed Principle
- **L**iskov Substitution Principle
- **I**nterface Segregation Principle
- **D**ependency Inversion Principle

Techniki

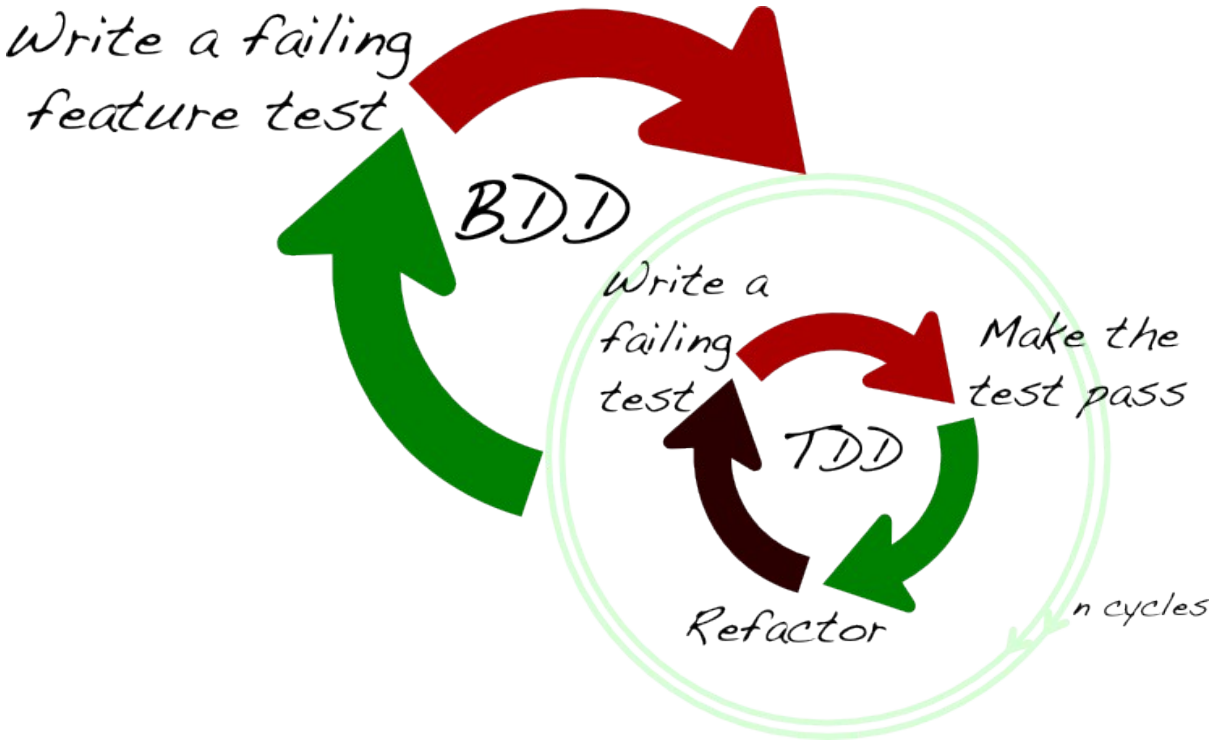
- **TDD – test-driven development**
- **BDD - behavior-driven development**

```
Story: Returns go to stock

In order to keep track of stock
As a store owner
I want to add items back to stock when they're returned

Scenario 1: Refunded items should be returned to stock
Given a customer previously bought a black sweater from me
And I currently have three black sweaters left in stock
When he returns the sweater for a refund
Then I should have four black sweaters in stock

Scenario 2: Replaced items should be returned to stock
Given that a customer buys a blue garment
And I have two blue garments in stock
And three black garments in stock.
When he returns the garment for a replacement in black,
Then I should have three blue garments in stock
And two black garments in stock
```



Techniki

- **TDD – test-driven development**
- **BDD - behavior-driven development**



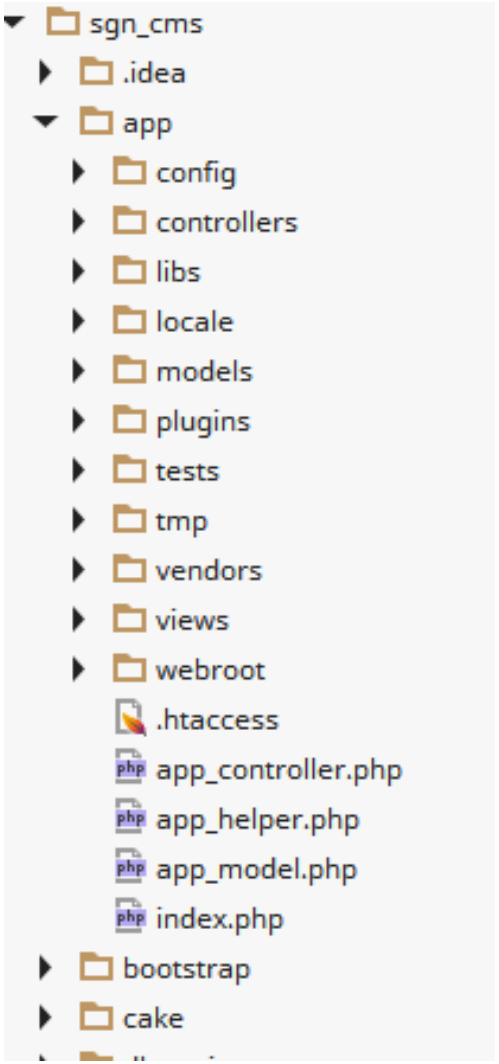
Architektura

- **MVC – model view controller**
- **MVVM – model view view-model**
- **MVP – model view presenter**

Wady MVC, MVVM, MVP

- **ODPOWIEDZIALNOŚĆ**
- **zależności**
- **baza danych (model) determinuje architekturę aplikacji**
- **bardziej złożone kontrolery zależą od wszystkich modeli**
- **gdzie jest miejsce logiki biznesowej?**

Logika biznesowa w centrum



- **Co wiadomo o tym projekcie?**
- **Co ta aplikacja potrafi?**

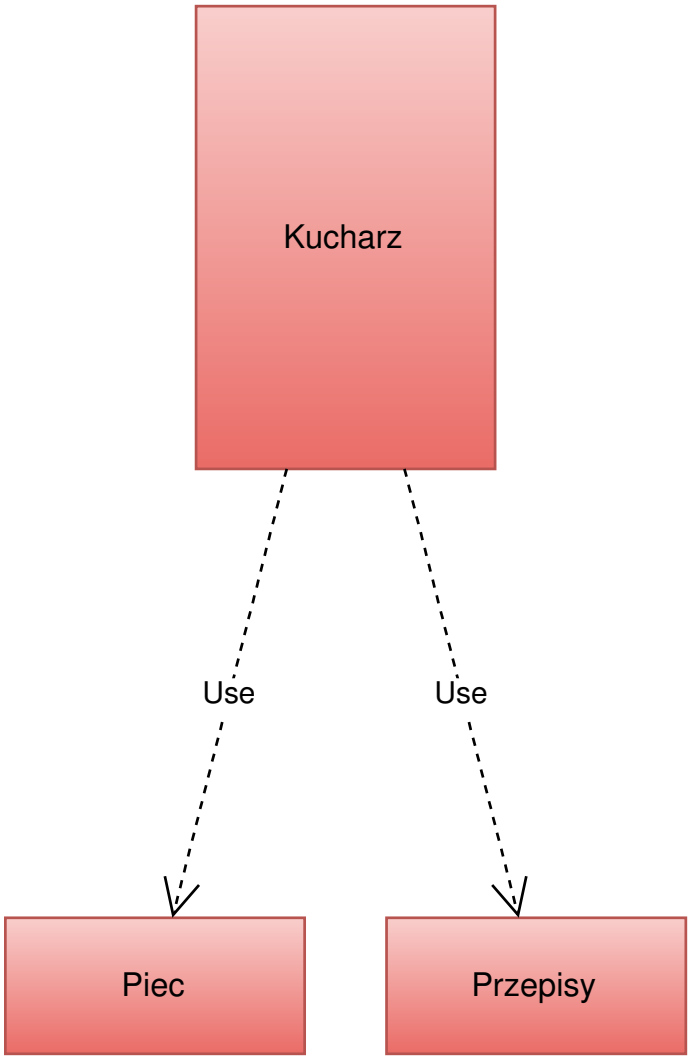
Clean Architecture

- spełnia zasady SOLID
- oferuje czytelny przepływ danych
- ułatwia testowanie
- podział na dane i klasy
- niemutowalne obiekty danych
- struktura aplikacji skupiona wokół logiki biznesowej

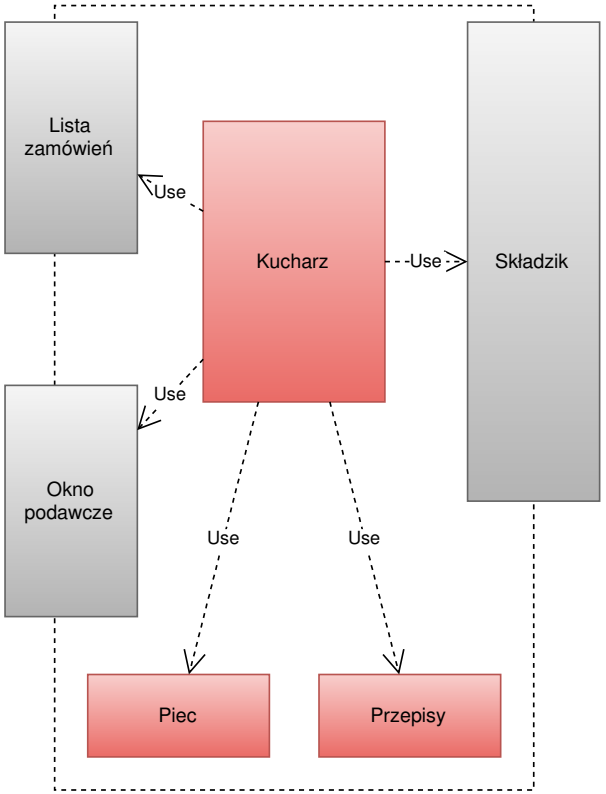
Clean Architecture - autor

- **Robert C. Martin**
- **Architecture the Lost Years:**
  https://www.youtube.com/watch?v=WpkDN78P884

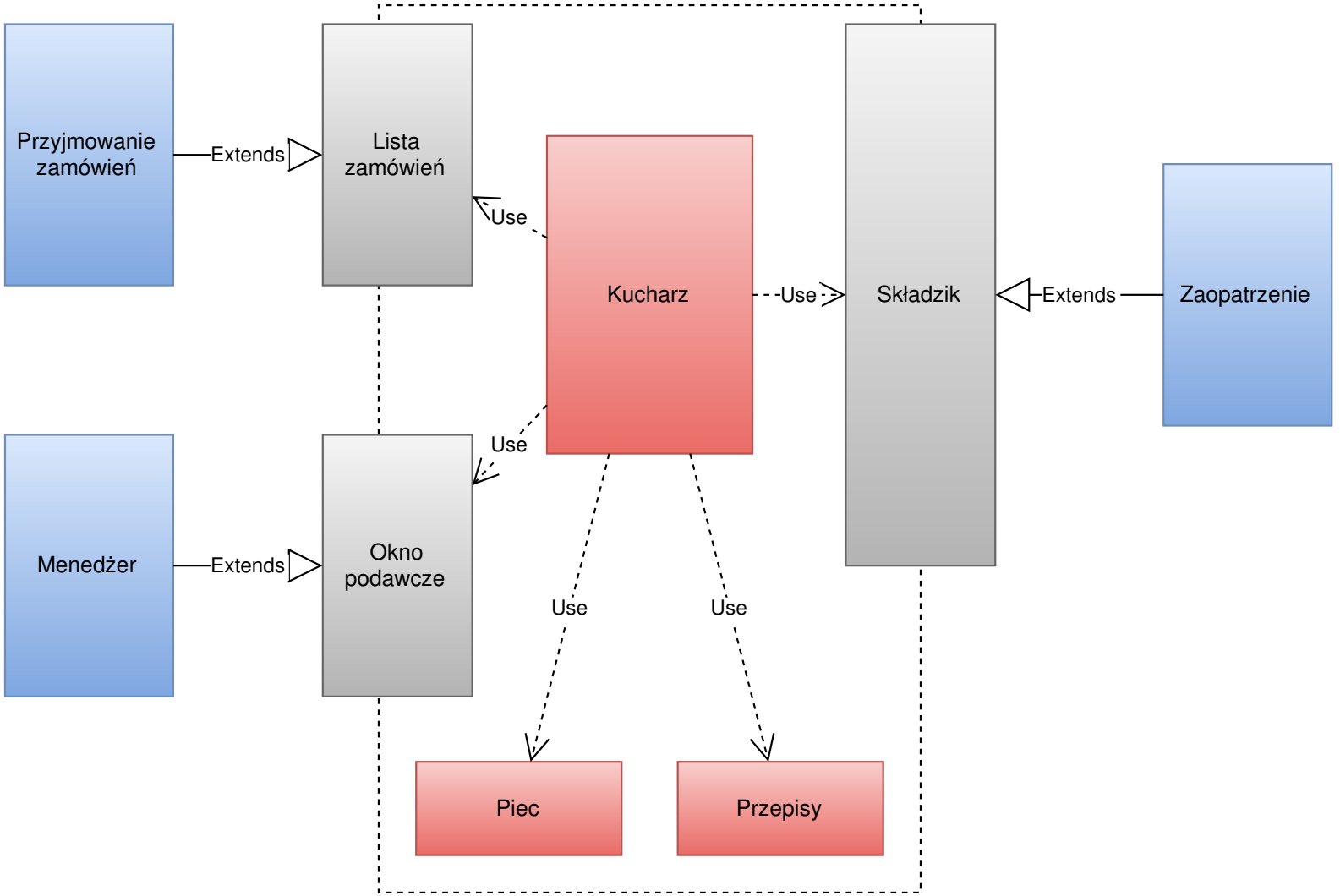
Pizzeria



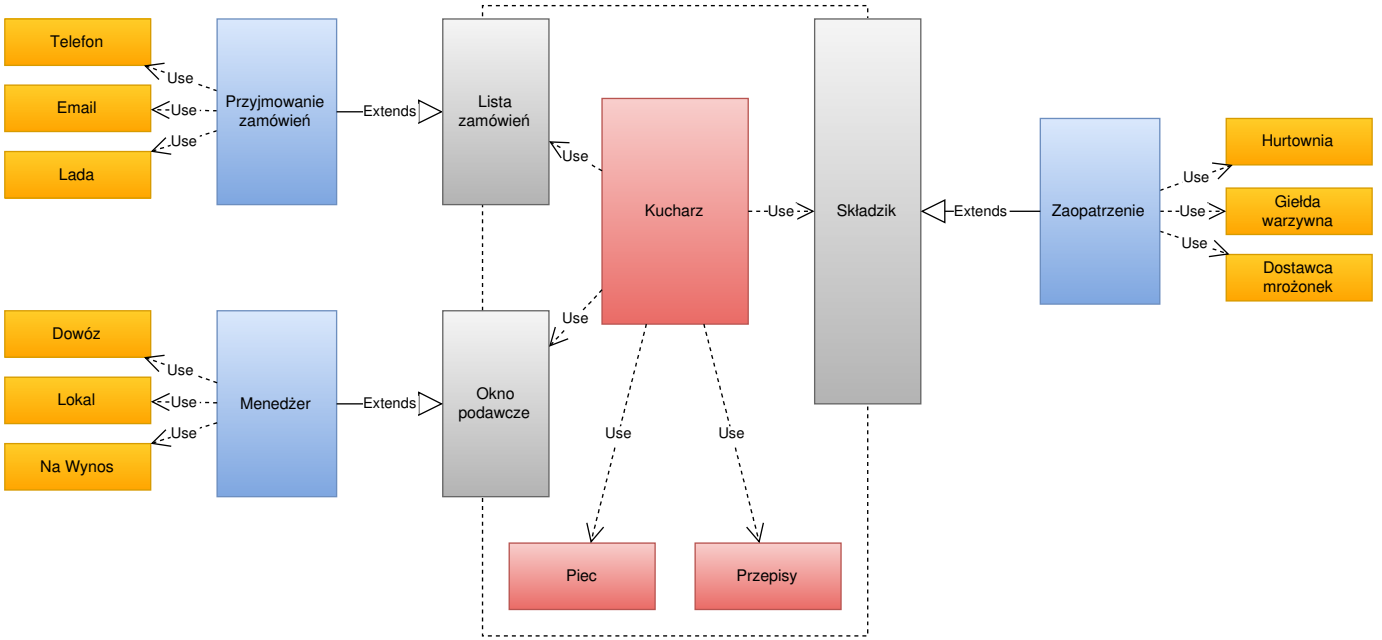
Pizzeria


Lista zamówień
Use
Kucharz
Use
Składzik
Use
Okno podawcze
Use
Use
Piec
Przepisy

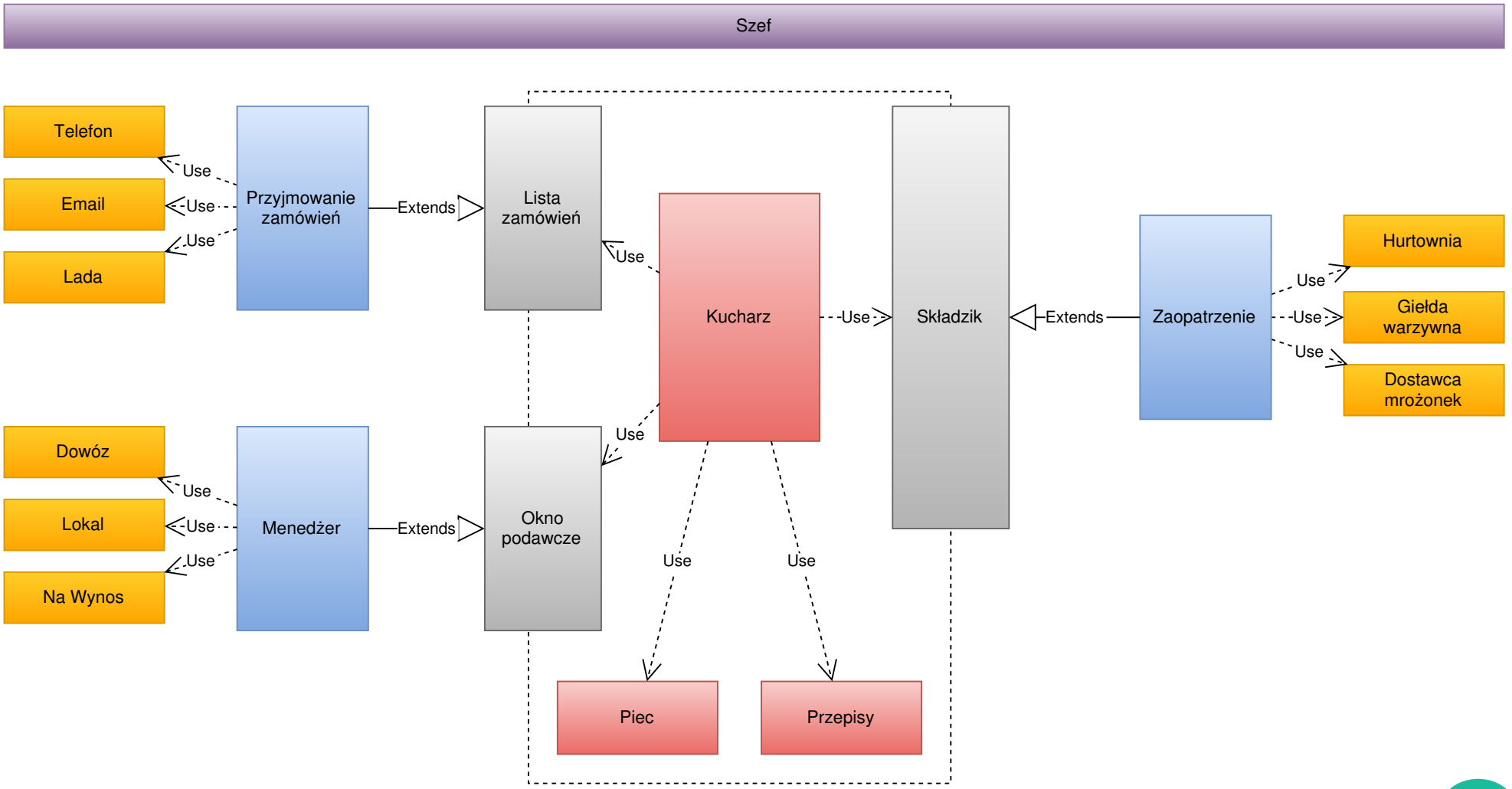
Pizzeria


Przyjmowanie zamówień
Extends
Lista zamówień
Use
Kucharz
Use
Składzik
Extends
Zaopatrzenie
Use
Menedżer
Extends
Okno podawcze
Use
Use
Piec
Przepisy

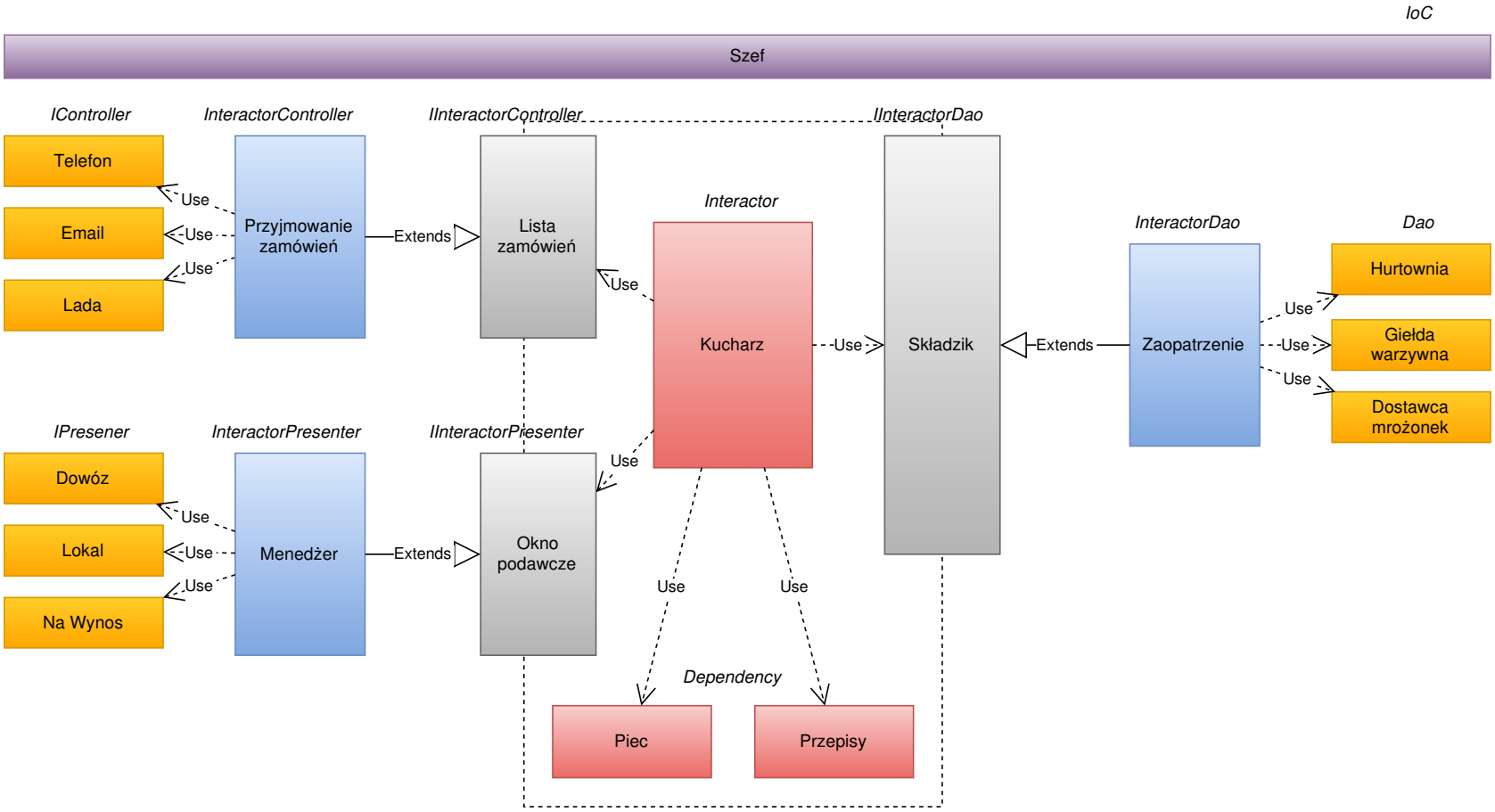
Pizzeria


Telefon
Email
Lada
Use
Use
Use
Przyjmowanie zamówień
Extends
Lista zamówień
Use
Kucharz
Use
Składzik
Extends
Zaopatrzenie
Use
Use
Use
Hurtownia
Giełda warzywna
Dostawca mrożonek
Dowóz
Lokal
Na Wynos
Use
Use
Use
Menedżer
Extends
Okno podawcze
Use
Use
Use
Piec
Przepisy

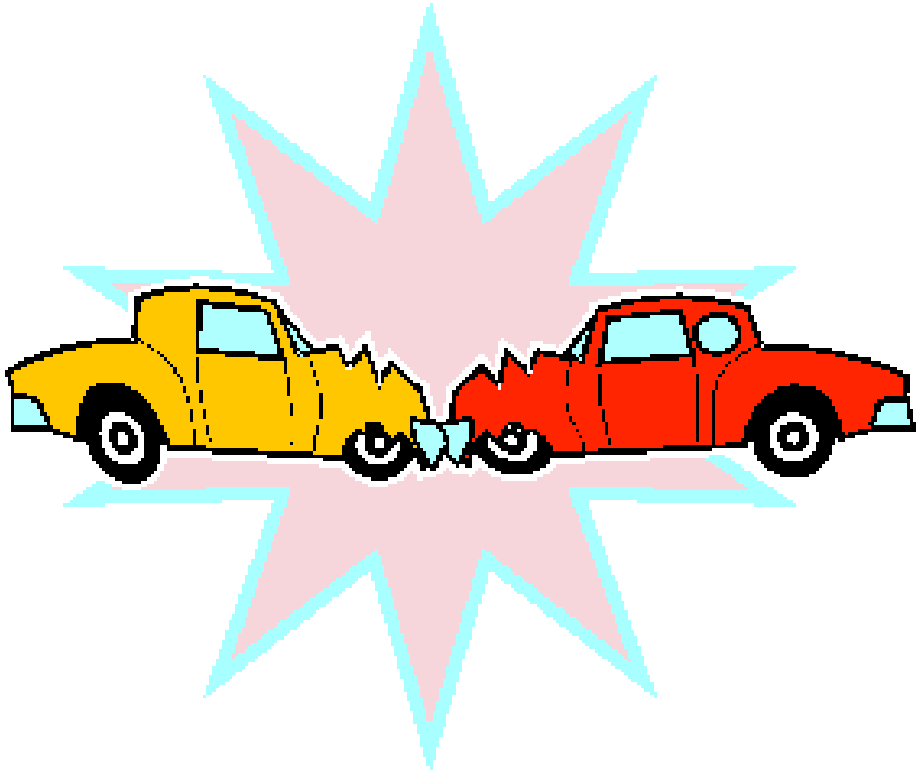
Pizzeria


Szef
Telefon
Email
Lada
Use
Use
Use
Przyjmowanie zamówień
Extends
Lista zamówień
Use
Kucharz
Use
Składzik
Extends
Zaopatrzenie
Use
Use
Use
Hurtownia
Giełda warzywna
Dostawca mrożonek
Dowóz
Lokal
Na Wynos
Use
Use
Use
Menedżer
Extends
Okno podawcze
Use
Use
Use
Piec
Przepisy

Pizzeria


IoC
Szef
IController
InteractorController
IInteractorController
IInteractorDao
Telefon
Email
Lada
Use
Use
Use
Przyjmowanie zamówień
Extends
Lista zamówień
Interactor
Use
Kucharz
Use
Składzik
Extends
InteractorDao
Zaopatrzenie
Dao
Hurtownia
Giełda warzywna
Dostawca mrożonek
Use
Use
Use
IPresener
InteractorPresenter
IInteractorPresenter
Dowóz
Lokal
Na Wynos
Use
Use
Use
Menedżer
Extends
Okno podawcze
Use
Use
Use
Dependency
Piec
Przepisy

Jak to połączyć?



- koszt wejścia
- brak doświadczenia
- błędy wynikające ze złego zrozumienia idei
- goniące terminy
- dług technologiczny*

* - http://wojciszko.com/2015/10/12/kiedy-warto-zaciagnac-dlug-technologiczny/

Własny framework

- A po co?
- Tego jest milion!
- Kolejny?!
- Czy będzie lepszy od istniejących?

NIE!!!

Po co?

- **dla nauki**
- **w projektach produkcyjnych nie ma czasu na naukę nowych rzeczy**
- **w projektach produkcyjnych nie można sobie pozwolić na niekończące się poprawianie kodu**
- **testowanie nowych rozwiązań**

BDD i TDD w praktyce

- błędy w BDD
- testowanie fabryk
- adaptacja bibliotek
- testowalna konfiguracja
- kolekcje w encjach biznesowych

Pytania?

Dziękuję za uwagę