

# **LEKCE 04**

Testování kódu

## Lekce 04 - Testování kódu

1. Co je to **Unit Testing**
2. Testovací Frameworky
3. Jak psát testy?
4. Jak unit testovat REST API controller pomocí **xUnit**
5. Mockování závislostí pomocí **NSubstitute**
6. Assertování pomocí **FluentAssertions**
7. Co je to Test Driven Development (**TDD**)
8. TDD v praxi

Naučíme se testovat kód, odhalíme benefity testování  
a zjistíme jak používat TDD metodiku testování ;)

**Pojďme otestovat tuto raketu**

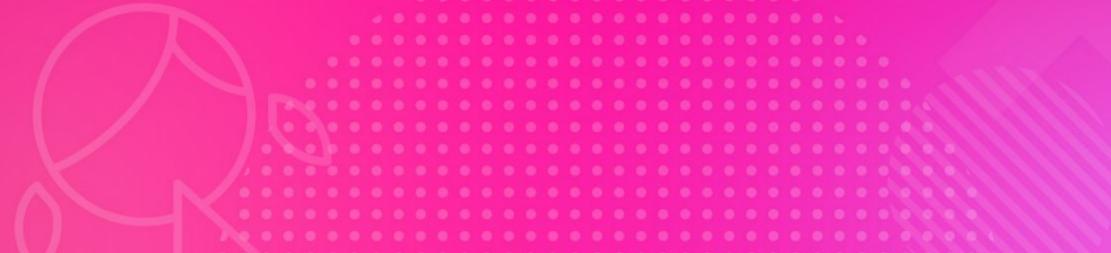


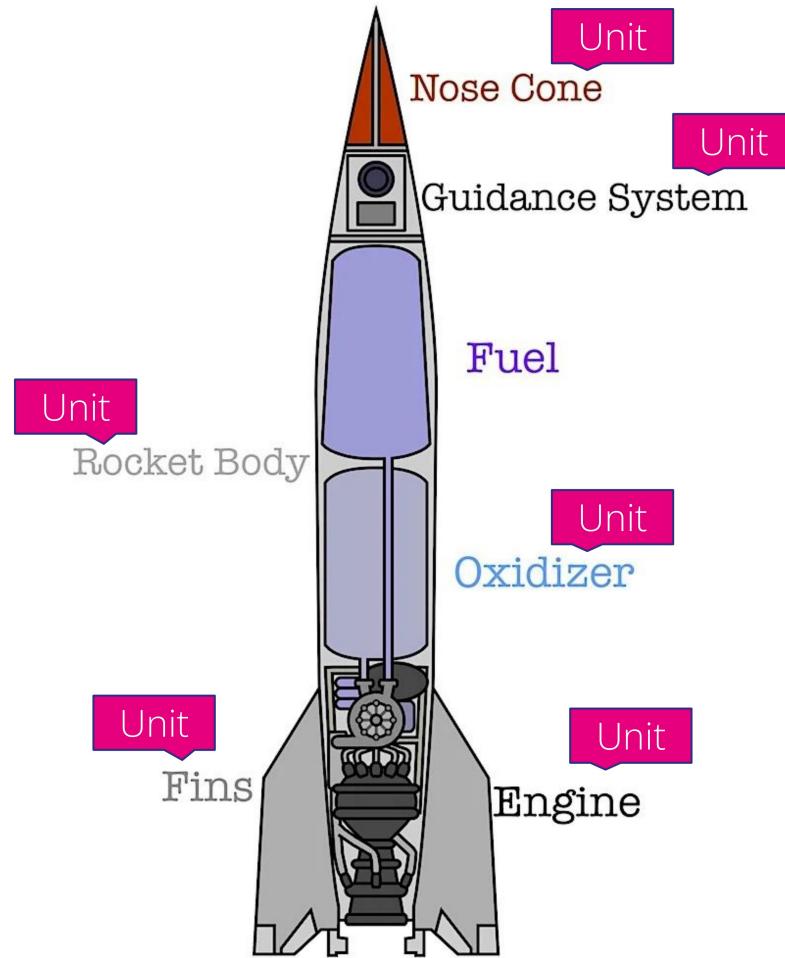
# **Co by se mohlo pokazit?**



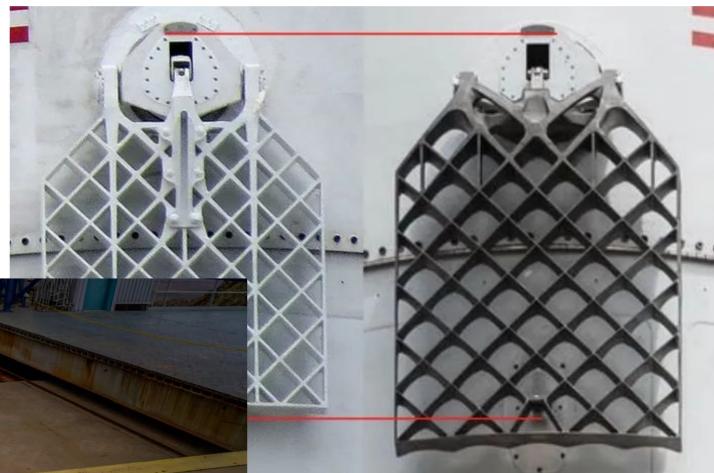


# Unit Testing





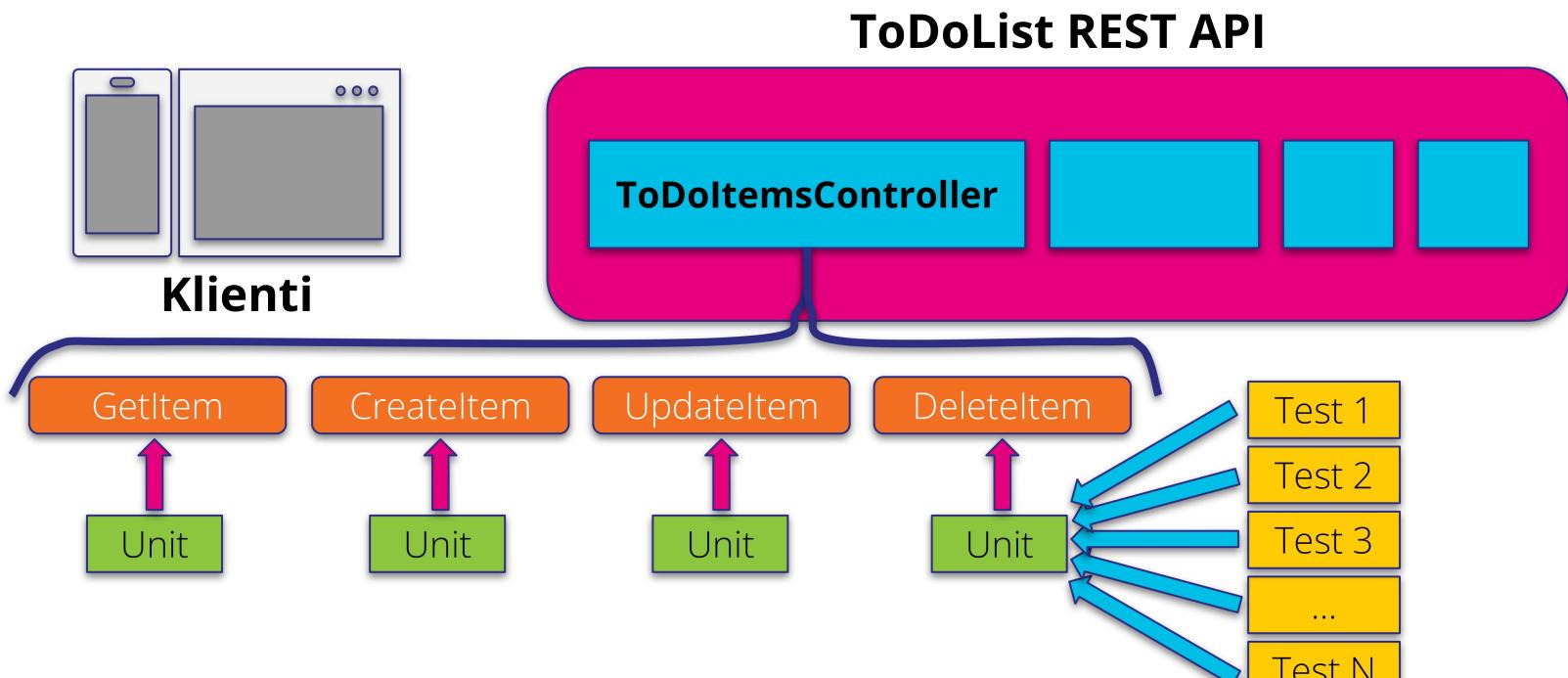
# Units



## **Co je to Unit Testing?**

*Unit Testing je testování kódu,  
kdy každá část kódu je testována v izolaci bez externích  
závislostí*

# Unit Testing REST API



## Proč unit testovat?

- Rychlá kontrola správnosti kódu
- Možnost dělat změny v kódu bez obav
- Snížení ceny opravy chyby (bugfixu)
- Živá dokumentace a specifikace

# Testovací frameworky



## **Unit testing frameworks v .NET**



**MSTest**

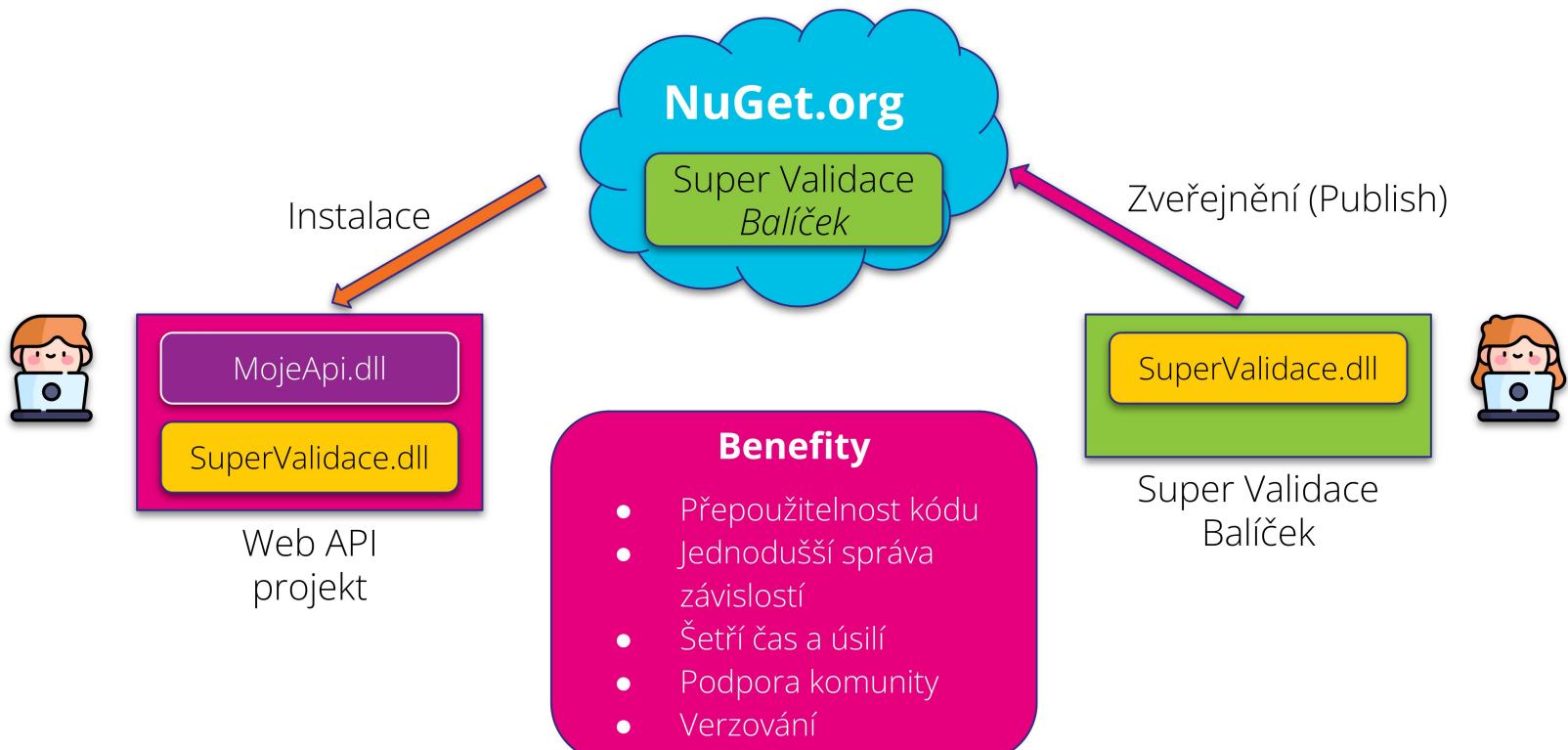
**xUnit.net**

Má  
preference

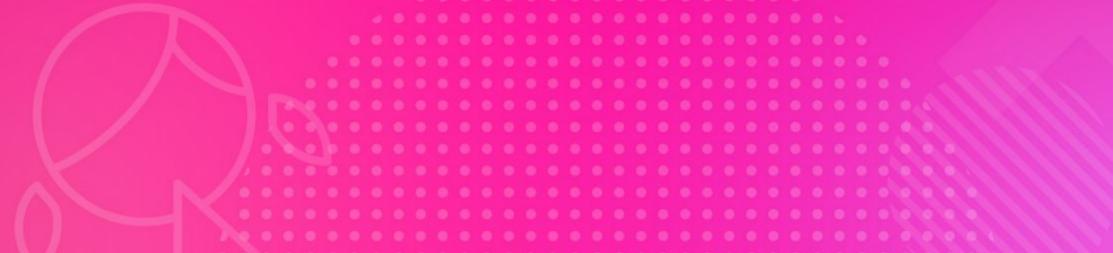
# NuGet



# Jak sdílet kód v .NET?



# Jak psát testy?



# **AAA pattern**

**A**rrange

Příprava kódu pro test

**A**ct

Vykonání akce, kterou chceme testovat

**A**ssert

Kontrola výstupu vykonané akce

## AAA pattern - ukázka v xUnit

[Fact]

```
public CalculatorSumReturnsCorrectSumOfThreeNumbers()
{
    // Arrange
    var calculator = new Calculator();
    // Act
    var result = calculator.Sum(1,2,3);
    // Assert
    Assert.Equals(result, 6);
    //result.Should().Be(6); //FluentAssertions
}
```

# **TDD**

## Test Driven Development

## **Co je to Test Driven Development (TDD)?**

*TDD je přístup k vývoji software, který je založen na psaní testů před napsáním logiky produkčního kódu.*

*Psaním testů definujeme funkcionality a chování systému, kterou následně implementujeme a napsanými testy ověřujeme.*

## Jak praktikovat TDD?

1. **Vytvoř nový test** reprezentující chtěnou funkčnosti/chování
2. **Spusť testy**, nový test by neměl procházet, všechny ostatní ano
3. Napiš **nejjednoduší možnou implementaci** tak, aby prošly všechny testy.
4. **Refactoring**

Dej kód kam logicky patří; Odstraňuj duplicity; Uprav názvy pokud neodpovídají kódu; Rozděl metody na menší;

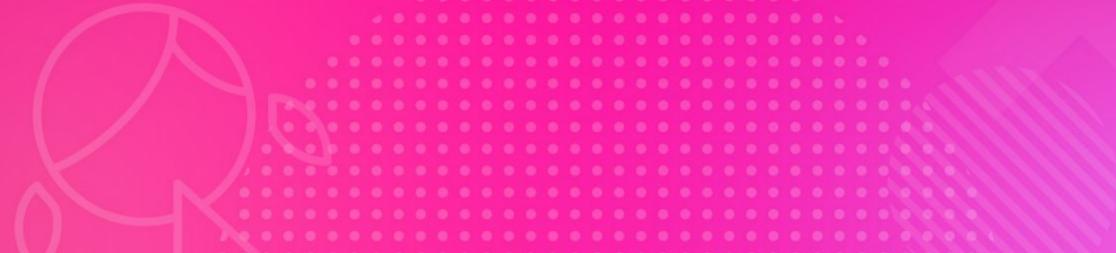
5. **Opakuj znovu od kroku 1**

## Proč praktikovat TDD?

- Vývoj je zaměřený na požadavky/funkcionalitu, né na implementaci
- Větší pokrytí produkčního kódu testy
- Chyby v kódu jsou díky testům odhaleny dříve
- Při změně požadavků se rychle dozvíme, co nefunguje.

**Všeho s mírou!**

# Mocking



## **Co je to Mockování?**

*Mockování je proces vytváření objektů (Mocků), které simulují chování reálných objektů.*

*Tyto Mocky lze v testovacím prostředí upravovat a nastavovat jejich chování.*

## **EXTRA: Typy zástupných objektů**

- **Dummy**
- **Stub**
- **Spy**
- **Mock**
- **Fake**

**Pro naše použití používáme Mocky!**

# Mockovací frameworky



# Mocking frameworks v .NET



Má  
preference



# **LEKCE 05**

Databáze

## Lekce 05 - Databáze

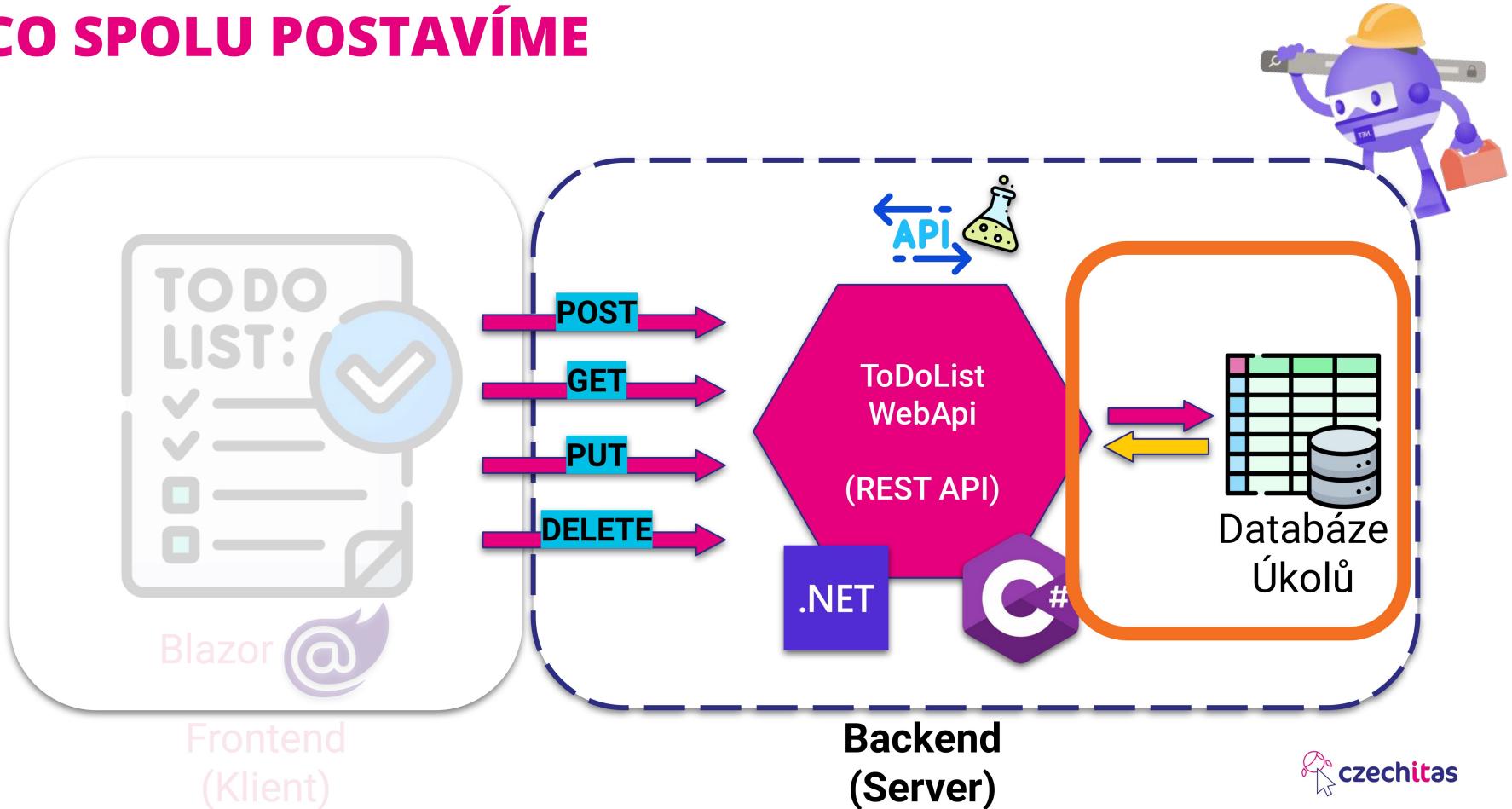
1. Co je to **databáze**?
2. **Tabulky, klíče, SQL**
3. Typy Databází
  - a. SQL
  - b. NoSQL
4. **O/RM**
  - a. **Entity Framework Core**

Seznámíme se s databázemi a SQL, víme co je to ORM a k čemu nám slouží.

Dokážeme se orientovat v základní práci s databázemi v projektu ;)



# CO SPOLU POSTAVÍME



# Database (DB)

## **Co je to databáze?**

*Databáze je organizovaná kolekce dat, která umožňuje efektivní uložení, načítání a správu informací.*

## Typické použití databází

- **Ukládání uživatelských dat** (např. e-shopy, webové aplikace)
- **Logistika** (sklady, přeprava)
- **Sledování transakcí** (banky, systémy e-commerce)
- **Skladování velkého množství dat** (analýzy, big data)
- **Blog** (články, komentáře, reakce)

# Jak vypadá taková databáze?

TABLE

id	name	category_id
1	Vytisknout fakturu	5
2	Zubař	4
3	Vyzvednout balík	3
4	Nakoupit	2
5	Vysávat	1

# COLUMNS



id	name	category_id
1	Vytisknout fakturu	5
2	Zubař	4
3	Vyzvednout balík	3
4	Nakoupit	2
5	Vysávat	1

# ROWS



# Keys

**Primary & Foreign**



## Primární klíč

### PRIMARY KEY



id	name	category_id
1	Vytisknout fakturu	5
2	Zubař	4
3	Vyzvednout balík	3
4	Nakoupit	2
5	Vysávat	1

id	name
1	Domácí práce
2	Jídlo
3	Pochůzky
4	Zdraví
5	Firemní

## Cizí klíč

### FOREIGN KEY



id	name	category_id
1	Vytisknout fakturu	5
2	Zubař	4
3	Vyzvednout balík	3
4	Nakoupit	2
5	Vysávat	1

id	name
1	Domácí práce
2	Jídlo
3	Pochůzky
4	Zdraví
5	Firemní

**TASKS**

PRIMARY		FOREIGN
id	name	category_id
1	Vytisknout fakturu	5
2	Zubař	4
3	Vyzvednout balík	3
4	Nakoupit	2
5	Vysávat	1

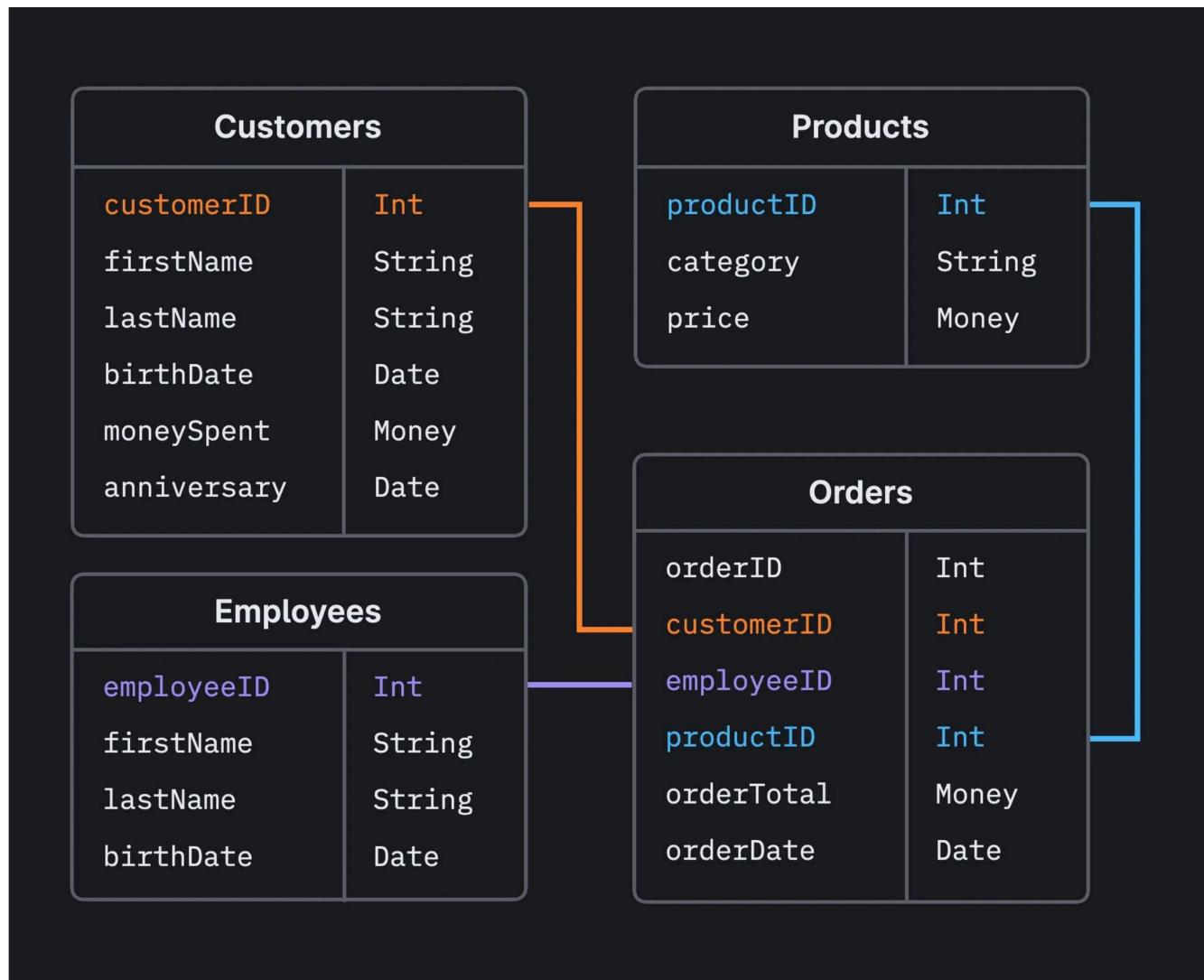
**CATEGORIES**

PRIMARY	
id	name
1	Domácí práce
2	Jídlo
3	Pochůzky
4	Zdraví
5	Firemní

## Jaké existují typy klíčů?

**Primary Key:** Unikátní identifikátor každého řádku v tabulce

**Foreign Key:** Sloupec, který odkazuje na primární klíč v jiné tabulce.

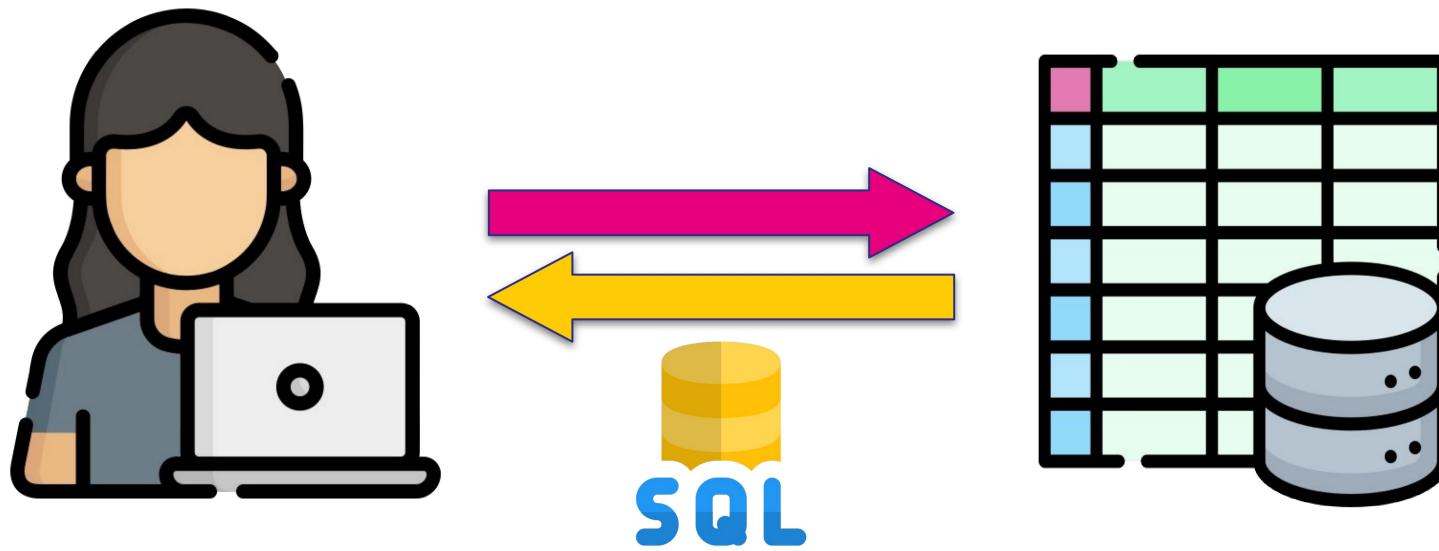


# **SQL**

## Structured Query Language

[sequel]

# Jak interagovat s databází?



## TASKS

### FOREIGN

id	name	category_id
1	Vytisknout fakturu	5
2	Zubař	4
3	Vyzvednout balík	3
4	Nakoupit	2
5	Vysávat	1

## CATEGORIES

### PRIMARY

id	name
1	Domácí práce
2	Jídlo
3	Pochůzky
4	Zdraví
5	Firemní



## Ukázka

Vyber všechny úkoly, jejichž kategorie je Domácí práce

```
SELECT * FROM Tasks WHERE category_id = 1;
```

## TASKS

### FOREIGN

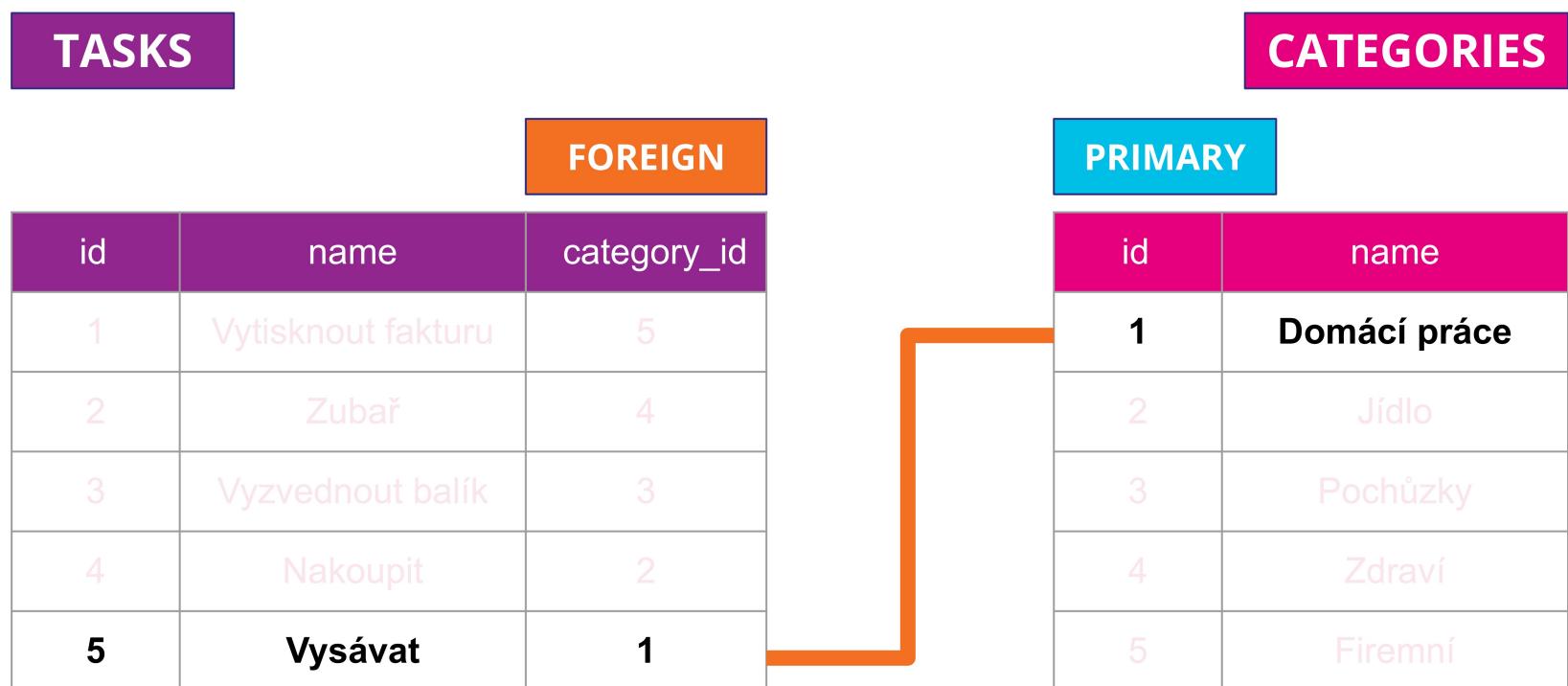
id	name	category_id
1	Vytisknout fakturu	5
2	Zubař	4
3	Vyzvednout balík	3
4	Nakoupit	2
5	Vysávat	1

## CATEGORIES

### PRIMARY

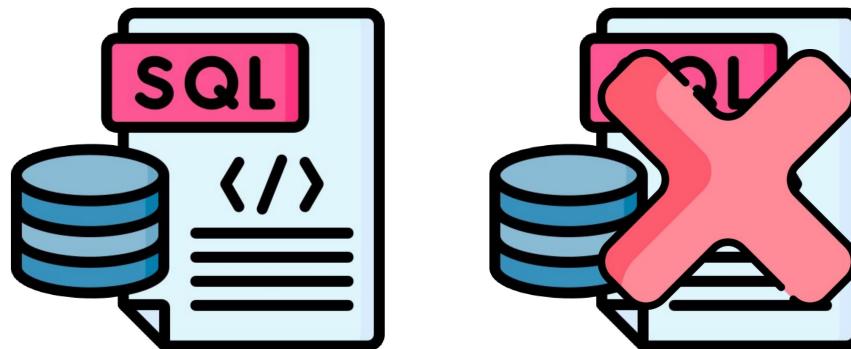
id	name
1	Domácí práce
2	Jídlo
3	Pochůzky
4	Zdraví
5	Firemní



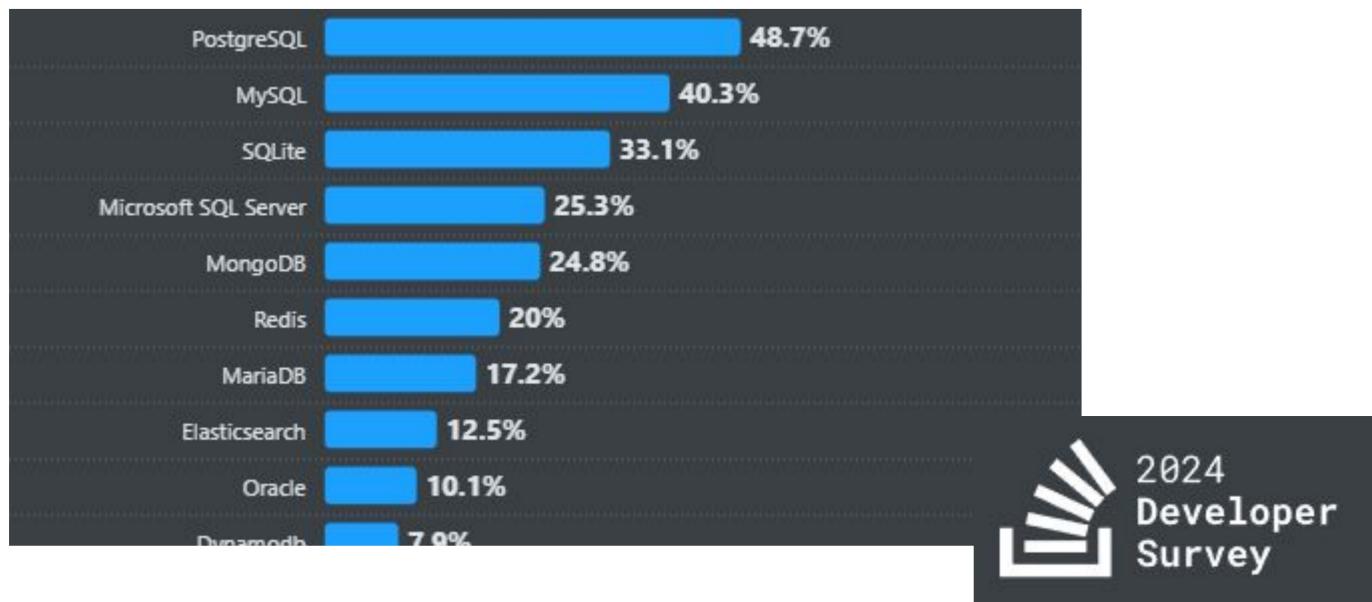


# **SQL vs NoSQL**

## V čem je ten rozdíl?



## Zastoupení na trhu



## Co použijeme v projektu?



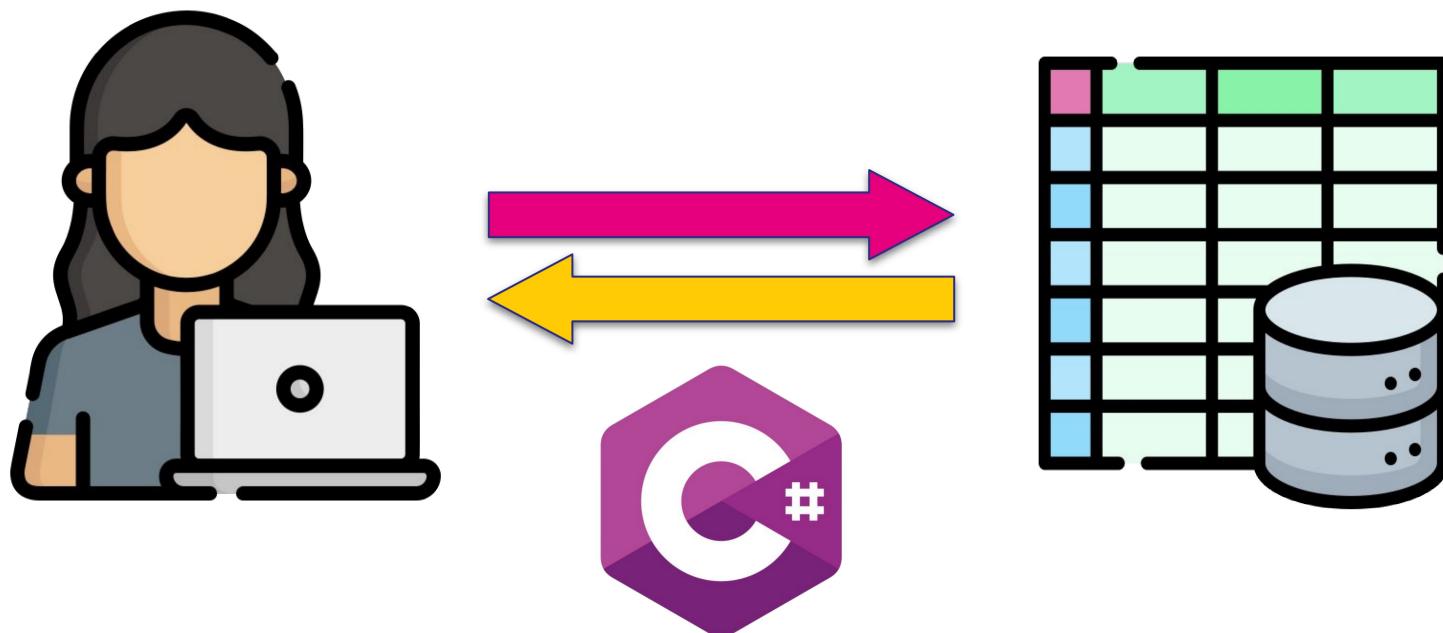
# O/RM

## Object-Relational Mapping

## **Co je to ORM?**

*Technika, která propojuje objekty z objektově orientovaného programovacího jazyka s databázovými tabulkami\**

# Jak interagovat s databází??



# Co použijeme?

Entity Framework



## Jak to vypadá podrobněji

