



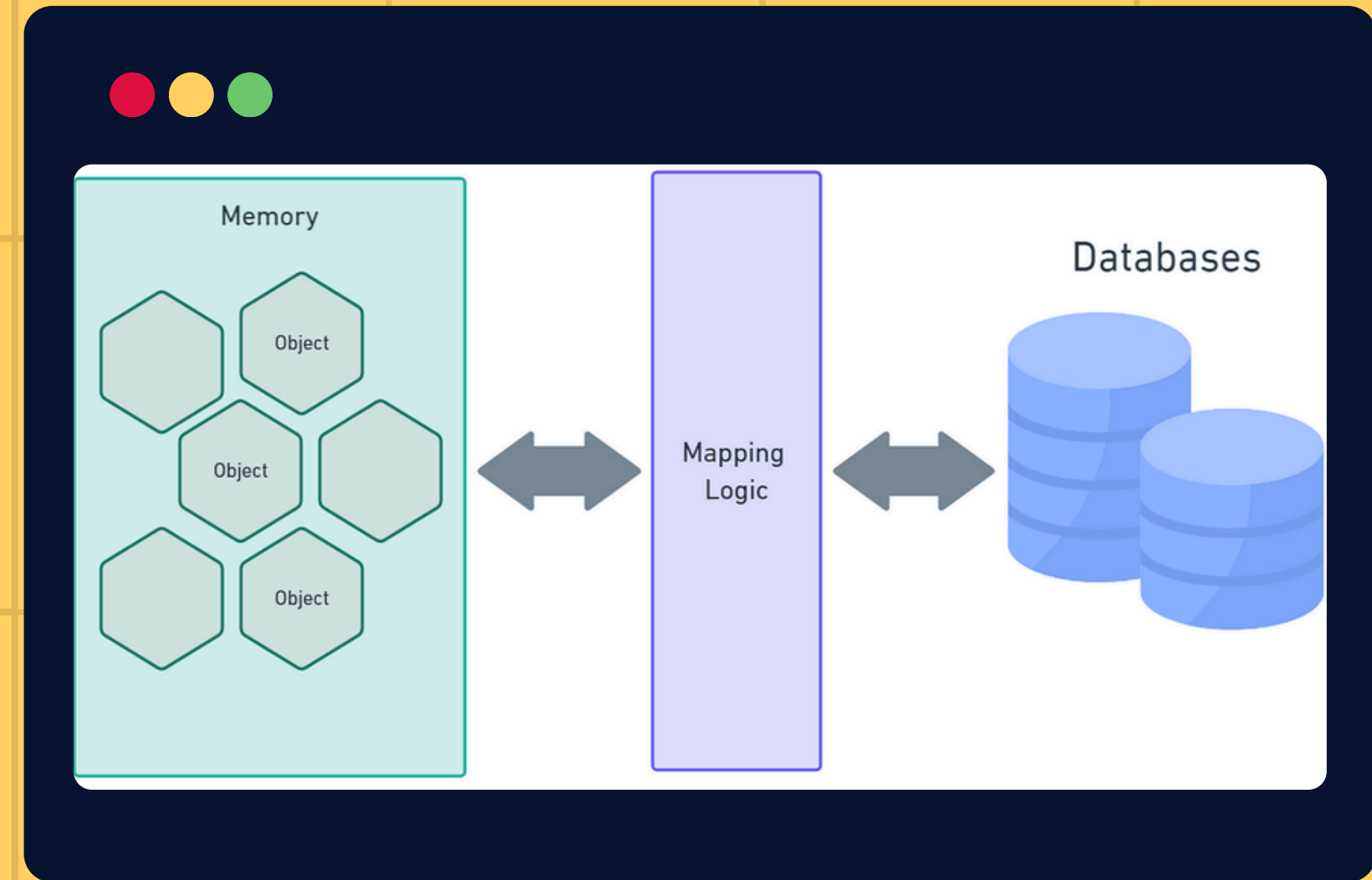
ORM & Entity Framework



Poklop
Kotek
Junková

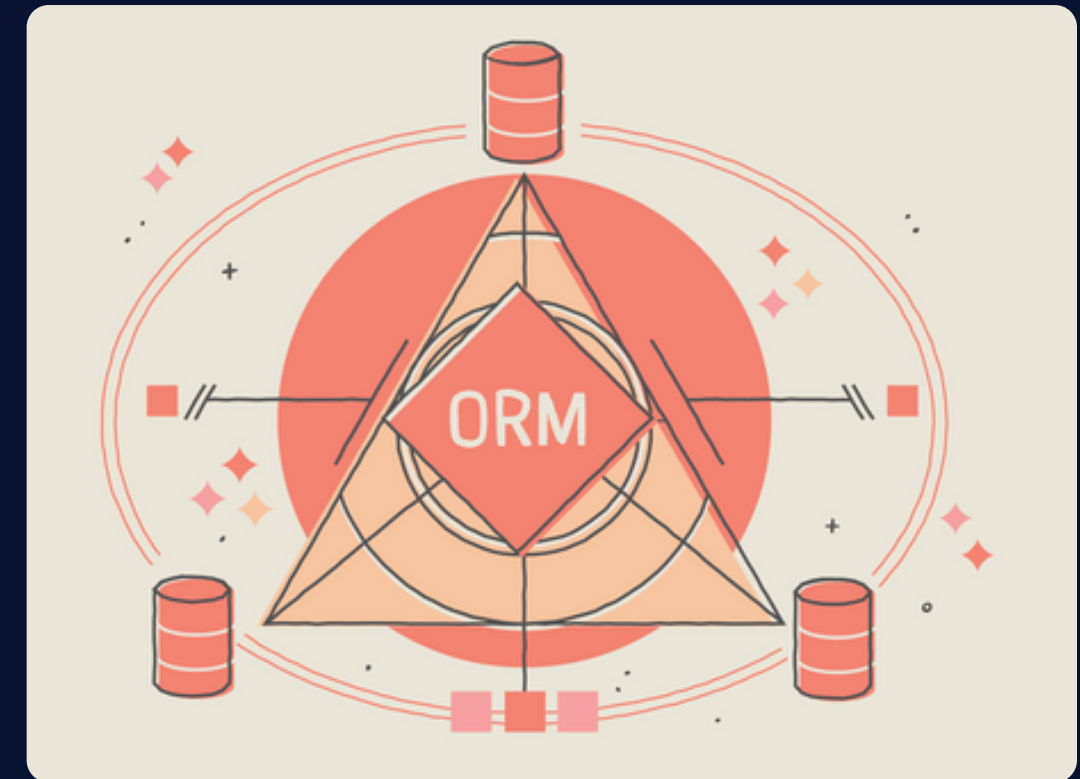
Co je ORM?

- ORM (Object-Relational Mapping) je technologie, která propojuje objektově orientované programování s relačními databázemi.
- Umožňuje programátorům pracovat s databázovými tabulkami pomocí objektů místo přímého psaní SQL dotazů.
- ORM funguje jako vrstva mezi aplikací a databází, která automaticky převádí objekty do tabulek a naopak.





ORM umožňuje mapování mezi objekty v kódu a tabulkami v relační databázi. Když aplikace potřebuje uložit objekt do databáze, ORM vygeneruje odpovídající SQL příkaz.



Jak funguje ORM?



Při načítání dat z databáze ORM převede tabulková data zpět na objekty. ORM také sleduje změny provedené na objektech a automaticky je synchronizuje s databází.

EF

```
1 using var context = new LibraryContext();
2
3 // CREATE
4 context.Books.Add(new Book { Title = "The Hobbit", Author = "J.R.R. Tolkien", Year = 1937 });
5 context.SaveChanges();
6
7 // READ & UPDATE
8 var book = context.Books.FirstOrDefault(b => b.Title == "The Hobbit");
9 if (book != null) { book.Year = 1951; context.SaveChanges(); }
10
11 // DELETE
12 context.Books.Remove(book);
13 context.SaveChanges();
```

BEZ ORM

```
1 using var conn = new SqlConnection("Server=...;Database=library;");
2 conn.Open();
3
4 // CREATE
5 var cmd = new SqlCommand("INSERT INTO Books (Title, Author, Year) VALUES (@t, @a, @y)", conn);
6 cmd.Parameters.AddWithValue("@t", "The Hobbit");
7 cmd.Parameters.AddWithValue("@a", "J.R.R. Tolkien");
8 cmd.Parameters.AddWithValue("@y", 1937);
9 cmd.ExecuteNonQuery();
10
11 // READ & UPDATE
12 cmd = new SqlCommand("UPDATE Books SET Year=1951 WHERE Title='The Hobbit'", conn);
13 cmd.ExecuteNonQuery();
14
15 // DELETE
16 cmd = new SqlCommand("DELETE FROM Books WHERE Title='The Hobbit'", conn);
17 cmd.ExecuteNonQuery();
```

Výhody & nevýhody ORM



Výhody

- Zjednodušení práce s databází
- Automatické mapování
- Bezpečnost
- Přenositelnost
- Zvýšení produktivity



Nevýhody

- Výkonová režie
- Složitější dotazy
- Učení se nové technologie

Populární ORM frameworky



- Python: SQLAlchemy, Django ORM
- C#: **Entity Framework**
- Java: Hibernate
- PHP: Doctrine

Entity Framework (EF)

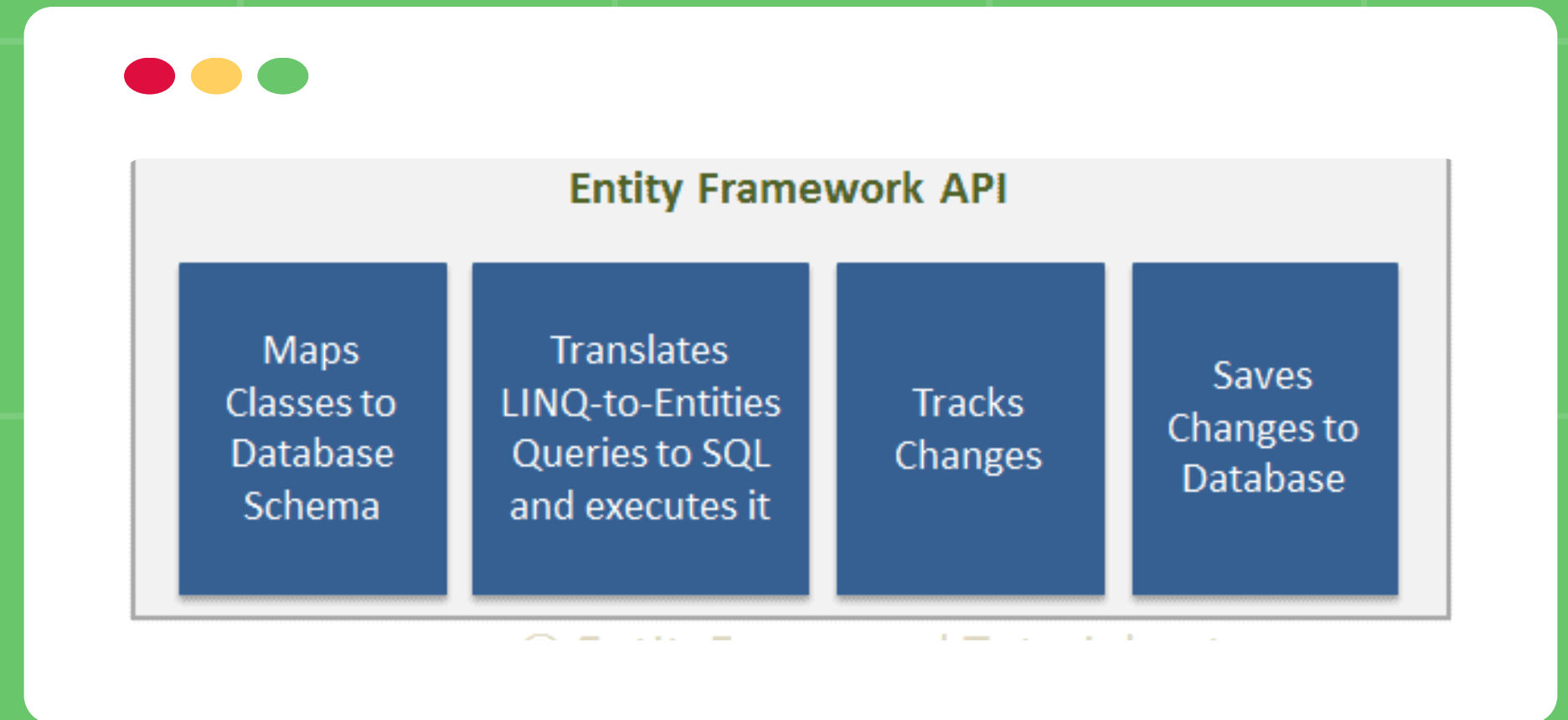
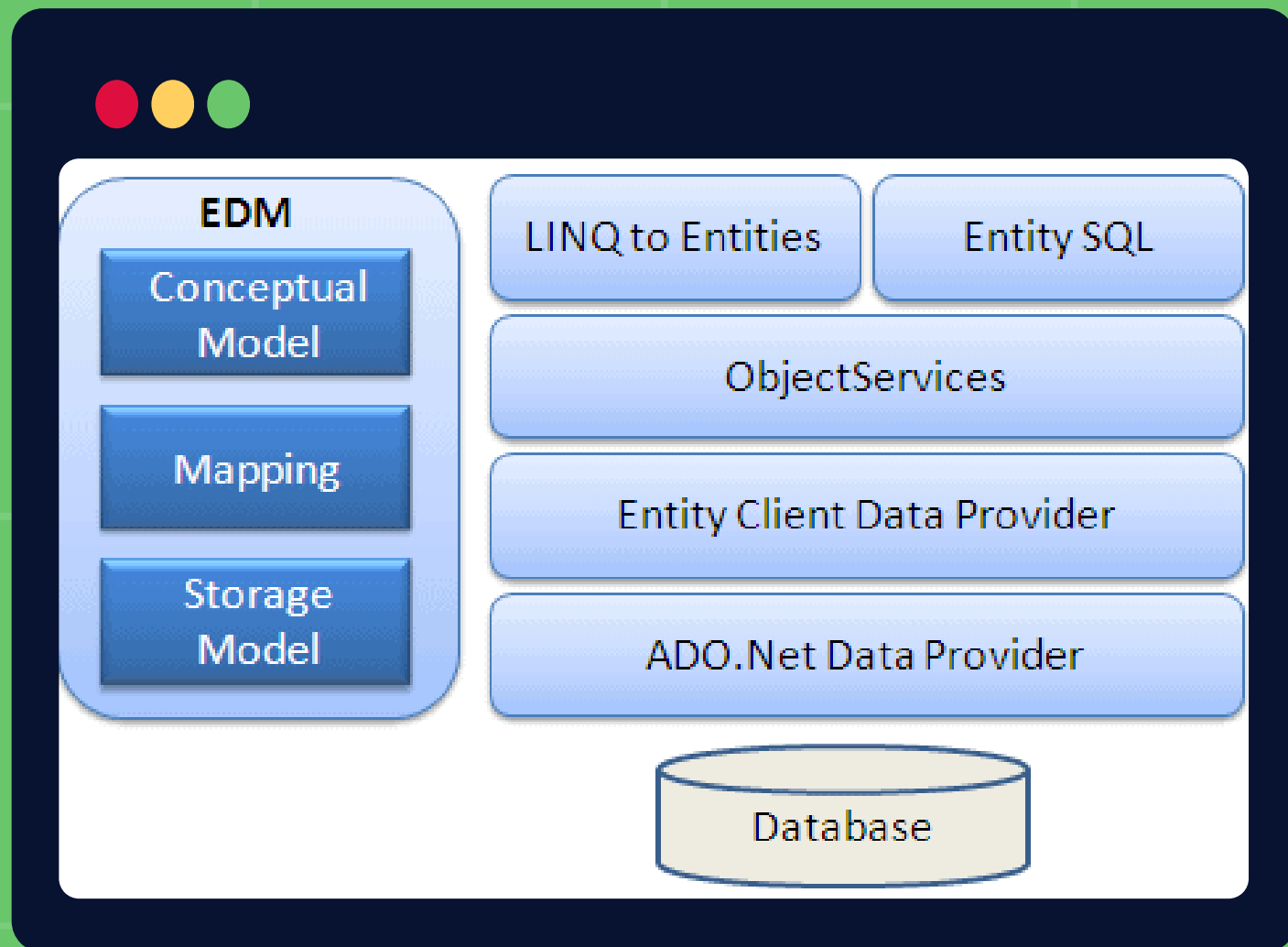


Entity Framework automatizuje databázové operace a umožňuje pracovat s daty jako s běžnými objekty v C#

- ORM (Object-Relational Mapping) framework pro C# a .NET
- Pracuje s databází jako s objekty (místo SQL dotazů)
- Automatizuje CRUD operace


```
1  using var context = new LibraryContext();
2
3  // CREATE
4  context.Books.Add(new Book { Title = "The Hobbit", Author = "J.R.R. Tolkien", Year = 1937 });
5  context.SaveChanges();
6
7  // READ & UPDATE
8  var book = context.Books.FirstOrDefault(b => b.Title == "The Hobbit");
9  if (book != null) { book.Year = 1951; context.SaveChanges(); }
10
11 // DELETE
12 context.Books.Remove(book);
13 context.SaveChanges();
```

Jak Entity Framework funguje?




- Vytváří datový model (EDM – Entity Data Model)
- Mapuje třídy na databázové tabulky
- Převádí dotazy LINQ na SQL
- Sleduje změny objektů
- Ukládá data pomocí `SaveChanges()`

Co je to Entita & Kontextová třída v EF?

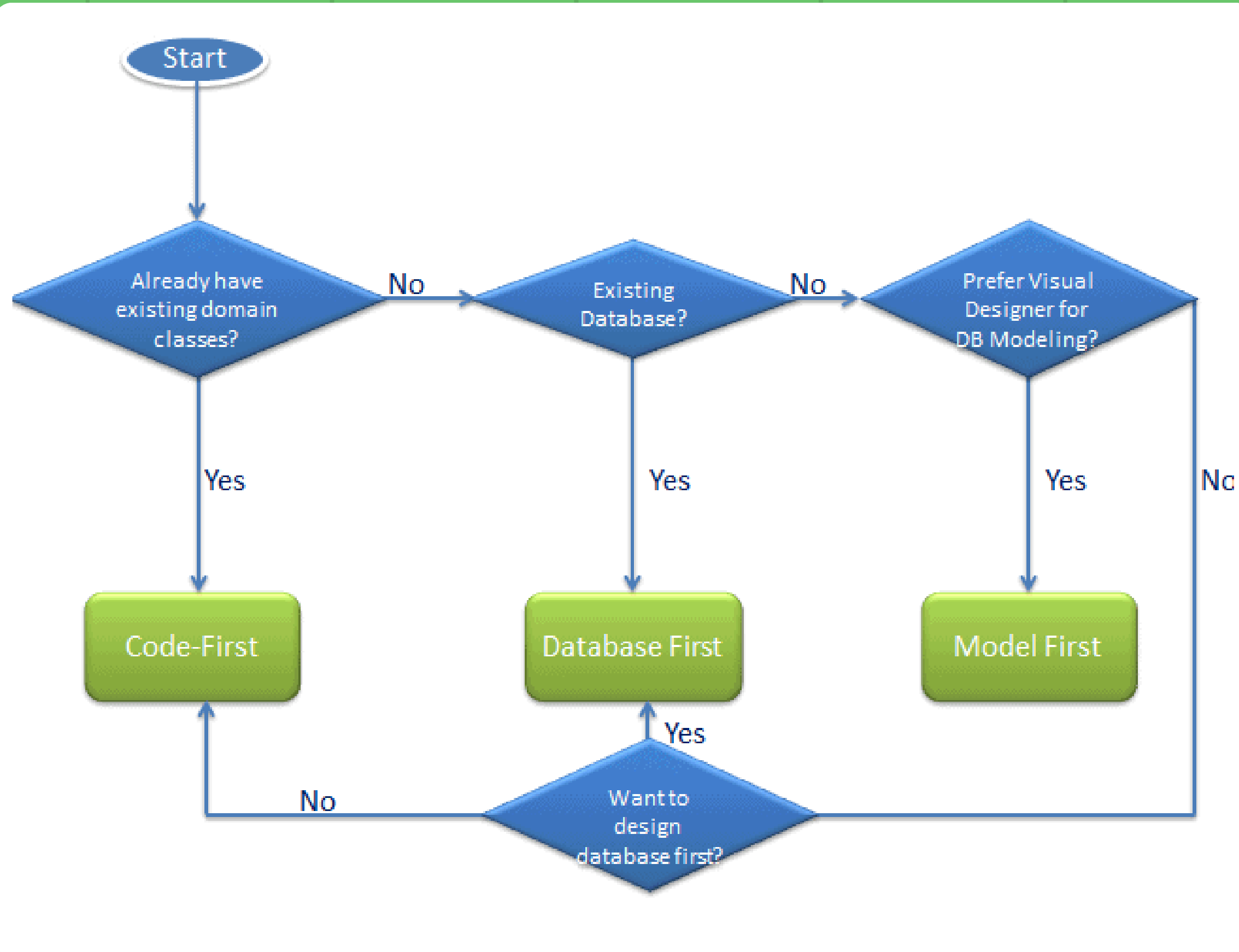


Entita = C# třída mapovaná na tabulku
Skalární vlastnosti = sloupce v databázi
Navigační vlastnosti = vztahy mezi entitami
DbSet = kolekce entit v kontextu
Stavy entit (Added, Modified, Deleted...)



Reprezentuje spojení s databází
Dědí z DbContext
Obsahuje DbSet<TEntity> – kolekce entit
Umožňuje provádět CRUD operace
Řídí transakce a změny objektů

Vývojové přístupy





**Děkujeme za
pozornost**