# Dekompozice softwarové logiky

## O co je jedná?

Jedná se o způsob rozdělení našeho projektu do více vrstev. Tyto vrstvy si můžeme představit jako jednotlivé projekty z nichž každý zastává pouze a jen svoji specifickou funkci.

## Proč je důležité správně dekomponovat náš projekt?

Správná dekompozice naší aplikace je kritická, protože eliminuje zbytečné komplikace při vývoji. Z pravidla platí, že čím rozsáhlejší aplikace je, tím robustnější musí být její struktura. Avšak, není na škodu se řádně nad strukturou zamyslet i u menších projektů, protože nikdy nemůžete s jistotou vědět, do jaké míry budete svoji aplikaci v budoucnosti potřebovat rozšířit.

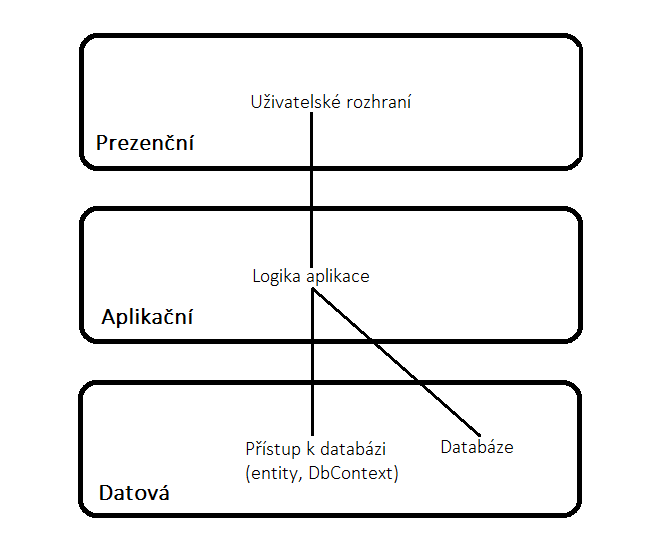
Špatná struktura naší aplikace může pro pozdější fázi vývoje znamenat její drastické zpomalení. Serióznost tohoto problému pak přímo úměrně stoupá s velikostí aplikace. Například, velké firmy mohou z důvodu špatné dekompozice softwarové logiky jejich projektu ztratit mnoho času a peněz, což často může vést i k bankrotu.

## Dekompozice aplikace na 3 vrstvy

Je mnoho způsobů, jak aplikaci rozdělit. Každý z těchto způsobů sebou přináší své výhody i nevýhody. My si popíšeme jeden z těchto způsobů, který spočívá v rozdělení projektu do 3 vrstev, tedy prezenční, aplikační a datové.

### Schéma

V obrázku můžete vidět, jak spolu jednotlivé vrstvy komunikují. Prezenční vrstva využívá aplikační vrstvu, která využívá datovou vrstvu. Tedy, prezenční vrstva ví o aplikační vrstvě, ale už neví o datové vrstvě.



### Popis jednotlivých vrstev

#### Prezenční vrstva

Prezenční vrstva poskytuje uživatelovi uživatelské rozhraní. Toto rozhraní uživatelovi umožňuje zobrazovat záznamy z databáze, vytvářet je a upravovat. Prezenční vrstva komunikuje s aplikační vrstvou. Projekt sloužící jako prezenční vrstva (webový projekt) tak musí mít referenci na projekt aplikační vrstvy (knihovna).

#### Aplikační vrstva

Aplikační vrstva obsahuje většinu logiky naší aplikace. V podstatě se stará o komunikaci mezi prezenční a datovou vrstvou. Zpracovává požadavky prezenční vrstvy a validuje je. Pokud projdou tyto požadavky validací, tak provede požadovanou operaci. Projekt aplikační vrstvy obsahuje referenci na projekt datové vrstvy.

Jako příklad může sloužit třeba vyhledání záznamu v databázi podle hodnoty, kterou zadal uživatel. Pokud hodnota splní dané požadavky, tak aplikační vrstva vytáhne data z databáze, entity převede na objekty jiného typu a vrátí je prezenční vrstvě. Objekty tohoto typu pak využije prezenční vrstva k naplnění modelů.

#### Datová vrstva

Datová vrstva slouží pro práci s databází. Její funkcí je tak načítat, vkládat a upravovat databázové záznamy. Tato vrstva jako jediná ze všech 3 nemá referenci na žádný jiný projekt, protože se nachází v nejspodnější části hierarchie.

<https://www.codeproject.com/Tips/733710/Layer-Architecture-for-Beginners-in-ASP-NET-Csharp>

<https://www.c-sharpcorner.com/UploadFile/4d9083/create-and-implement-3-tier-architecture-in-Asp-Net/>

* obecně (co to je)
* rozdělení na 3 vrstvy
  + prezenční
  + business
  + datová