Desenvolvimento de Aplicações Empresarias

# O que faz uma aplicação ser empresarial?

* **Longo ciclo de vida**
  + Uma aplicação empresarial tem um longo ciclo de vida, que por muito esforço que façamos para a matar a mesma nunca morre, como exemplo temos o AGCP ou mesmo o código em Cobol que faz funcionar os bancos, apesar dos anos que o mesmo tem, ele continua a funcionar.
* **Muitos Ecrãs**
  + Abre várias janelas que mostram informação, clico num artigo, vou ler sobre o artigo, clico noutro link para saber mais e vai abrindo várias janelas.
* **Multicanal**
  + A aplicação tem mais de uma maneira de interagir com o utilizador, tanto seja por computador, telemóvel, tablet, etc.
* **Integração com outras aplicações / sistemas de informação**
  + Uma aplicação empresarial não é solitária, também está dependente de outras aplicações, por exemplo uma aplicação de compra de um artigo, está dependente da de faturação, outra para transporte, etc.
* **Muitos utilizadores**
  + Uma aplicação empresarial tem vários utilizadores, muitos mais que nós possamos imaginar, como exemplo o Facebook.
* **Muitos dados e muitos dados para persistir**
  + Uma aplicação empresarial tem vários dados, que são a base do funcionamento da aplicação, e nós não queremos que o utilizador ao fechar a aplicação ou que a mesma vá abaixo, que todos os dados sejam perdidos.
* **Regras de negócio (muitas e/ou complicadas)**
  + Um negócio tem várias regras que têm de ser seguidas, assim como as suas aplicações que estão a ser desenvolvidas.
* **Indispensáveis aos funcionamentos da empresa/organização**
  + Uma aplicação nunca pode falhar ou ir abaixo, tem de estar sempre ativa 100%, como exemplo quando o sistema de multibanco do continente foi abaixo, toda a gente deixou as suas compras e foi se embora.

Apenas necessita-se que um dos pontos esteja a ser utilizado, para a aplicação passar a ser considerada uma aplicação empresarial.

No telemóvel, temos várias aplicações que utilizamos, mas não são empresariais, como exemplo os jogos que temos ou a aplicação da calculadora. Como exemplo não é por uma aplicação ser desenvolvida em Java que é uma aplicação empresarial.

Apenas escolhemos as tecnologias que vamos utilizar, depois de sabermos que requisitos é que ela requer, e não ao contrário.

O portal do estudante é uma aplicação empresarial, pois integra-se com outras aplicações.

Text, letter

Description automatically generated

# Arquitetura de software & modelo 3 camadas

## Definições

* Organização de alto nível e estrutura (relações) de componentes do sistema
* Princípios (regras) de desenho
* Decisões fundamentais (difíceis de alterar posteriormente)
* Fundação para o desenvolvimentos e evolução do sistema
* Suporta e implementa funções de negócio
* Deve permitir a evolução do sistema (estrutura, requisitos, funcionalidade)
* Existem várias arquiteturas (ou vistas) no sistema.

## Para que serve

### Perspetiva organizacional

* Comunicação do desenho de alto nível
* Fornece contexto ao sistema
* Alocação de trabalho (dividir a arquitetura de trabalho, dividir a equipa de back-end e front-end, para que cada pessoa ou componente funcionem independentemente)

### Perspetiva técnica

* Cumprir requisitos e objetivos
* Potencia flexibilidade
* Potencia redução de custos de manutenção e evolução
* Aumento a reutilização e integração com sistema legacy e componentes de terceiros

## O que não é arquitetura de software

* Modelo de dados
* Arquitetura física onde o sistema vai executar
* Detalhes de desenho/implementação

## Exemplos de modelos padrões

## Modelo de 3-layer (3 camadas)

Padrão porque já foi testado praticamente em vários casos, tornando num dos modelos bases para soluções.

O sistema é dividido em 3 camadas separadas:

* Serviços de apresentação (PLL)
  + Fornecem a interação entre o utilizador (ou sistema externo) e a aplicação
* Regras de negócio (BLL)
  + Representam o núcleo da aplicação em termos de processamento
* Serviços de dados (DAL)
  + Fornecem serviços de persistência

Mais sobre:

* Normalmente cada camada está num componente isolado
* Cada camada apenas depende da camada seguinte
  + Normalmente não são admitidos “saltos” por cima de uma camada (ex: da 1º para a 3º)
* Permite alterações em qualquer uma das camadas sem interferência nas outras
  + Desse que se mantenham as interfaces das classes
* Potencia separação de responsabilidades
* Permite especialização da equipa de desenvolvimento
* Potencia a facilidade de manutenção do código
* Pode aumentar a complexidade de compreensão do código

## A 1º camada: Apresentação

* Apenas se preocupa com a interação com o utlizador
* Possui lógica de apresentação
  + Tratamentos de inputs
  + Visualização de resultados
  + Validações que aumentem a usabilidade
* Pode ser um programa não interativo (batch)
* Permite implementar a filosofia multi-canal

## A 2º camada: lógica de negócio

* Apenas se preocupa com a implementação das regras de negócio associadas ao problema e às entidades de negócio
* Desconhece “quem” é a aplicação cliente
* Implementa todas as validações de dados que necessitar
* Desconhece os pormenores de persistência dos dados
  + Recorre aos serviços da camada de de acesso a dados.

## A 3º camada: Serviços de dados

* A 3º camada não são os dados, nem os servidores de base de dados, mas sim os serviços de acesso aos dados
* Encapsulas, de um ponto de vista de operações de negócio, o acesso aos dados, isolando a camada de lógica de negócio
* Strings de SQL só devem existir nesta camada
* Tecnologias tipo ADO.net ou JDBC não são a 3º camada, poderão ser consideradas uma 4º camada