



IPS Instituto
Politécnico de Setúbal
**Escola Superior de
Tecnologia de Setúbal**

Programação Avançada 2021-22

Padrão de Arquitetura - MVC

Bruno Silva, Patrícia Macedo

Sumário



- Padrão **MVC** (Model-View-Controller)
 - Enquadramento
 - Problema
 - Solução Proposta (pelo padrão)
 - Exemplo de Aplicação
 - Exercícios

Motivação 🤔 | Contexto Histórico

O aparecimento do MVC resultou da evolução tecnológica, que levou a uma crescente necessidade distribuição das componentes por distintas máquinas físicas.

- Aplicação desenvolvida para ser usada numa única máquina:



- Base de Dados partilhada pelos vários clientes:

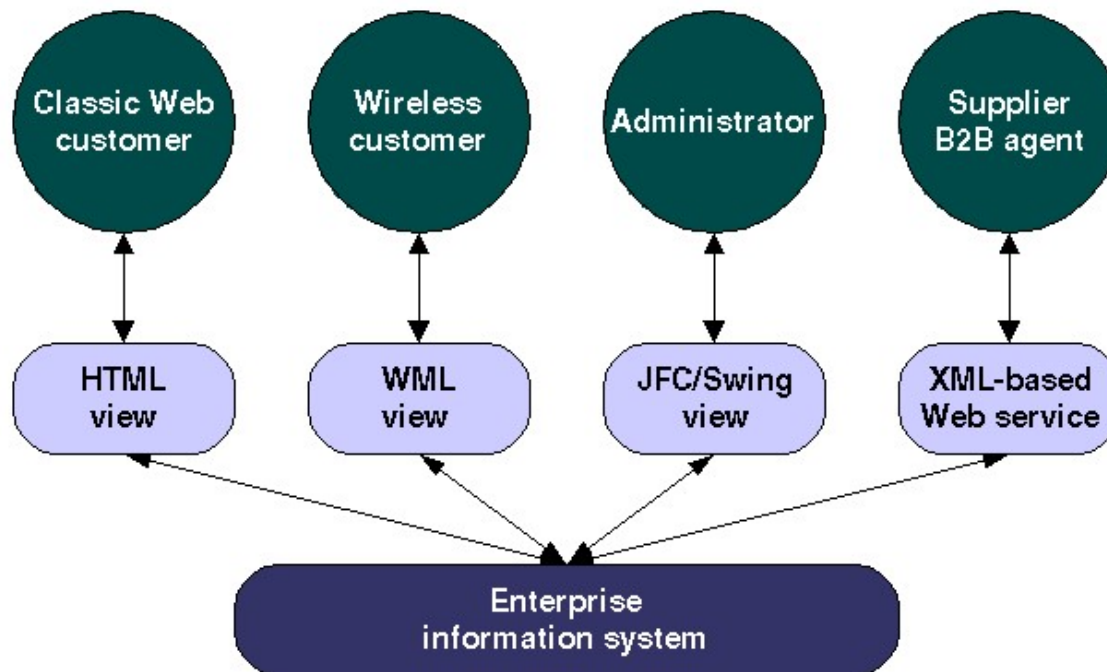


- Servidor de Aplicações partilhado pelos vários clientes Web:

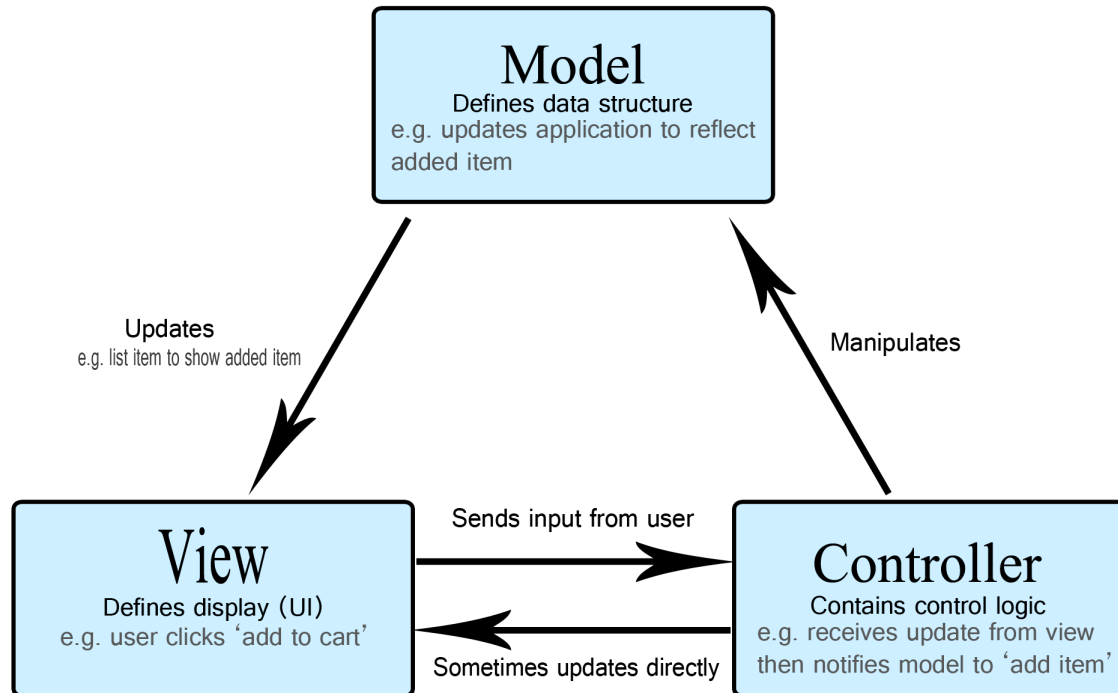


Motivação 🤔 | Contexto Histórico

A existência de diferentes tipos de utilizadores, com diferentes tipos de interfaces que acedem à mesma informação, levou à necessidade da existência de uma arquitetura que suportasse múltiplas interfaces.



Solução Proposta | Padrão MVC



💡 Separa as funcionalidades principais do negócio, da apresentação e da lógica de controle. Utiliza o padrão **Observer** (view -> model).

Solução Proposta | Padrão MVC

Esta separação permite partilhar os mesmos dados por diferentes vistas/*user interfaces*. E torna mais fácil as tarefas de desenvolvimento, teste e manutenção para os diferentes *clientes*.


Participantes | Padrão MVC

MODEL

- 🧐 Sabe tudo sobre:
 - Os dados persistentes que devem ser apresentados;
 - As operações que serão aplicadas para transformar os dados.
- 🧑 Nada sabe sobre:
 - As interfaces do utilizador;
 - Como os dados serão mostrados;
 - As acções das interfaces usadas para manipular os dados.

Participantes | Padrão MVC

VIEW

- Conhece o *modelo* (participante MODEL), mas **não o manipula** (apenas consulta);
- Define como os dados serão visualizados pelo utilizador;
- Mantém consistência na apresentação dos dados quando o *modelo* muda;
-  **Não implementa lógica de negócio.**
- Delega ao *controlador* (participante CONTROLLER) aos ações/intenções do utilizador;

Participantes | Padrão MVC

CONTROLLER

- Sincroniza as ações do utilizador com a manipulação do *modelo*;
- Implementa lógica de controle e validação.
- Pode manipular a *vista* (participante VIEW) diretamente;

Enquadramento

O MVC é um padrão largamente utilizado no desenvolvimento de aplicações *web* e *mobile*, estando embebido em diversas frameworks:

- Kendo
- Angular JS
- React
- [ASP.NET](#) MVC
- Spring Framework

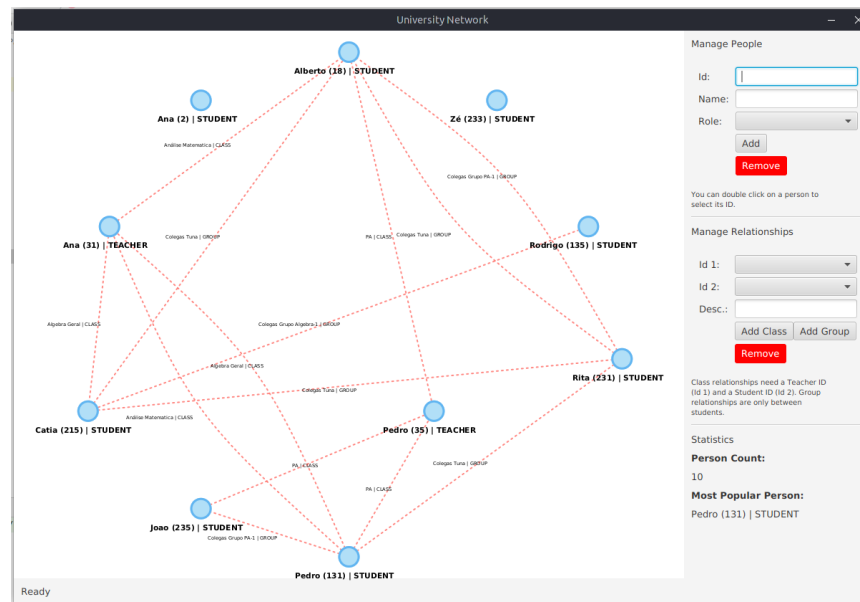
Existem também "variantes" deste padrão, e.g., *MVP* (Model-View-Presenter) e *MVVM* (Mode-View-ViewModel) que modificam ligeiramente as responsabilidades dos participantes.

Exemplo de aplicação

Repositório de apoio à aula no GitHub:

https://github.com/estsetubal-pa-geral/JavaPatterns_MVC

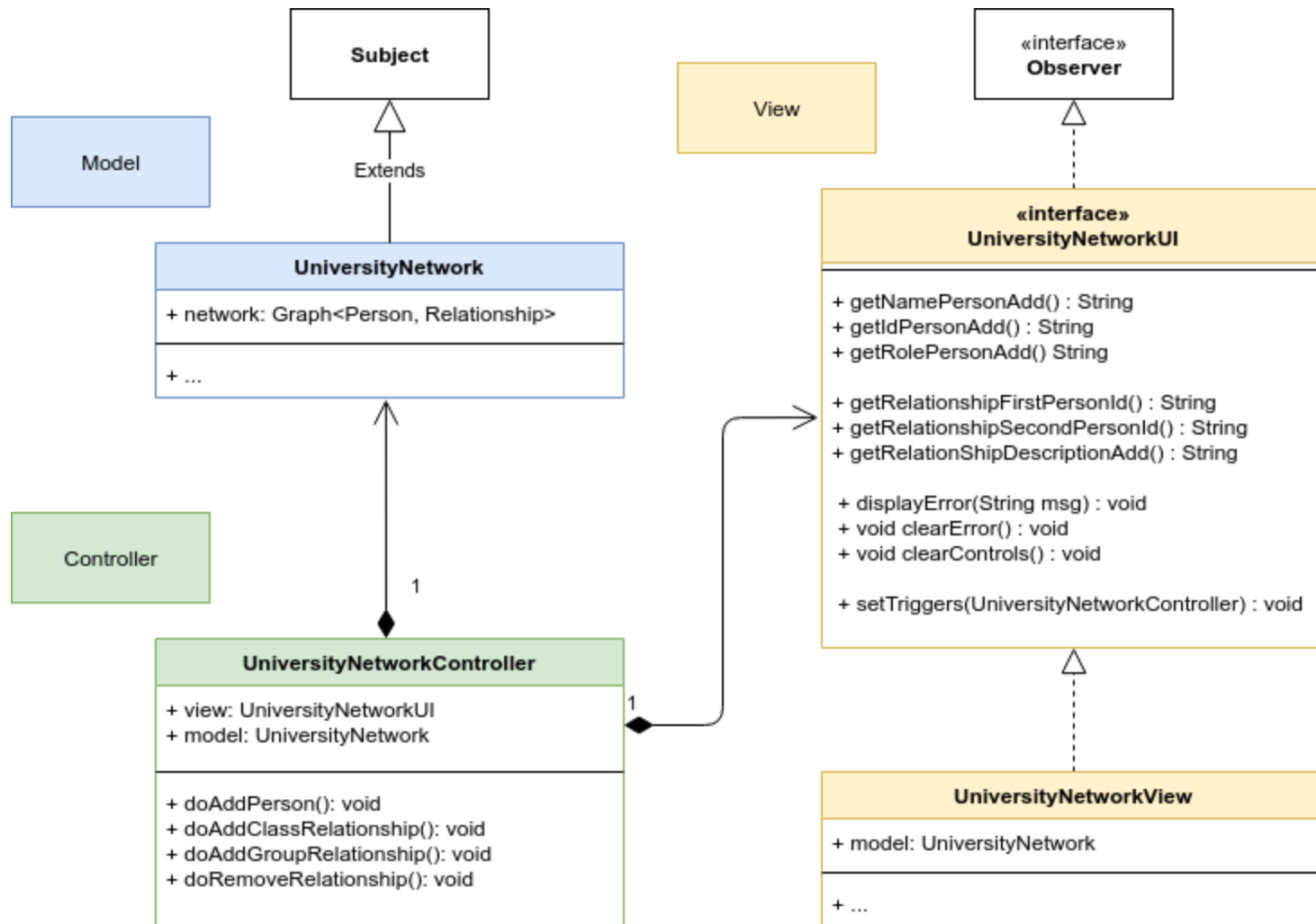
Apresenta-se uma aplicação *JavaFX* desenvolvida com o padrão MVC:



Extensão de trabalho laboratorial anterior - *University Network*.

Exemplo de aplicação

A arquitetura da aplicação é a seguinte:



Exemplo de aplicação

- Optou-se por definir a *interface* `UniversityNetworkUI` que descreve todas as operações passíveis de serem efetuadas sobre uma *user interface* (view);
 - O *controlador* sabe que operações pode invocar sobre a *view*;
 - Isto permite que se possam criar *user interfaces* alternativas, mantendo a mesma estrutura de classes.
- A *view* manter-se-á atualizada com o *model* através da utilização do padrão **Observer**;
- O método `setTriggers` será invocado pelo *controller* para que a *view* possa "delegar" os *user inputs* para as ações do controlador, e.g., `doAddPerson()`.

Exercícios

1. Execute a aplicação e visualize o funcionamento inicial;
 - Analise os construtores dos participantes e o *main*.
2. Forneça o código necessário para as "*Statistics*" serem populadas corretamente.
3. Adicione código necessário para **adicionar uma nova pessoa** à rede.
 - Fornecer código em `setTriggers` da *view* e `doAddPerson` no *controller*.
4. Implemente a funcionalidade de **remover uma pessoa da rede**.
 - Necessário implementar código no *model*, *controller* e *view*.

Leitura adicional

Outras perspectivas sobre o MVC

https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

<https://realpython.com/the-model-view-controller-mvc-paradigm-summarized-with-legos/>

<https://www.educba.com/javascript-mvc-frameworks/>