

# Aplicações Multi-camada

2019/20

Contribuições de:

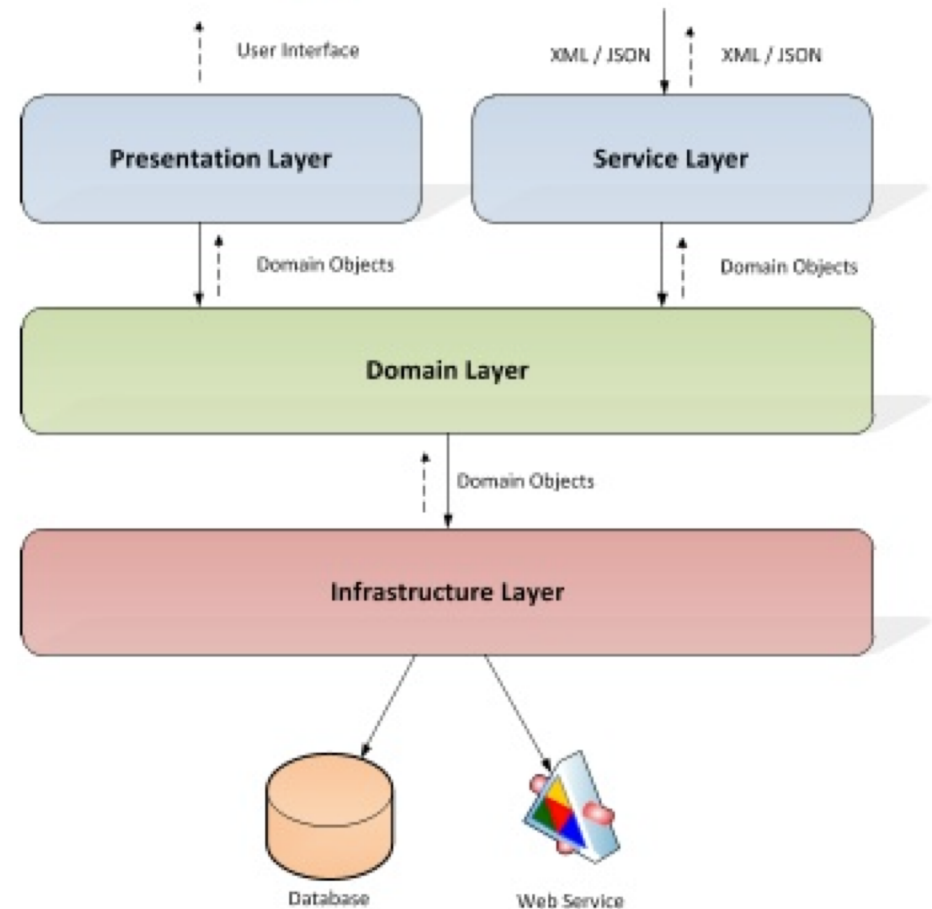
António Nestor Ribeiro

José Creissac Campos

Rui Couto

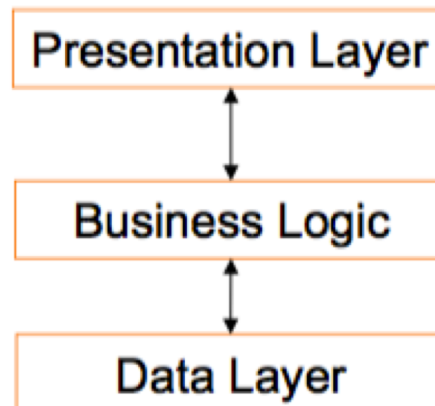
# Aplicações Multi-Camada

- Diversos tipos de objectos para efectuarem diferentes operações
  - Presentation Objects
  - Business/Domain Objects
  - Data Objects



## Três Camadas

- A camada de interface, **Presentation Layer**, permite isolar a interface com o utilizador por forma a que o resto da aplicação esta não esteja dependente de uma interface concreta.
- A camada de negócio, **Business Logic**, implementa a lógica da aplicação.
- A camada de dados, **Data Layer**, permite isolar o acesso aos dados, por forma a que o resto da aplicação não esteja dependente da origem ou estrutura sob a qual os dados estão armazenados.

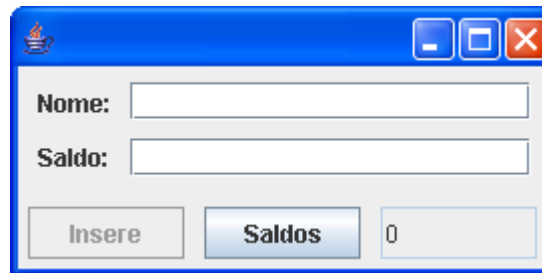


# Camada de Apresentação

- Programação Orientada ao Evento
  - Controle da aplicação pode estar na camada de interface
  - Aplicação limita-se a responder a eventos:
    - clicar do rato num botão
    - inserir um caractere num campo de texto
    - ...
  - Durante a inicialização da aplicação são registados os métodos que serão chamados quando ocorrerem determinados eventos.

# Camada de Apresentação

- Programação Orientada ao Evento
  - Que eventos podemos/temos de tratar neste exemplo?

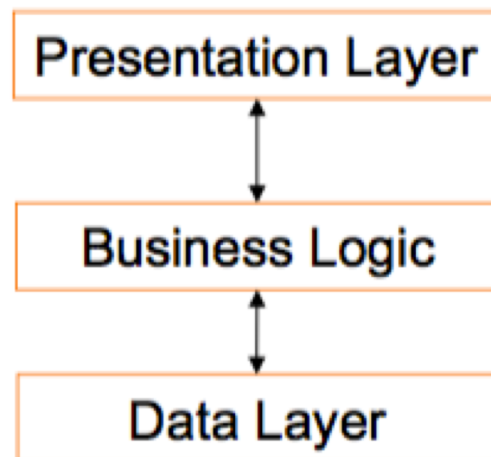


Nome:

Saldo:

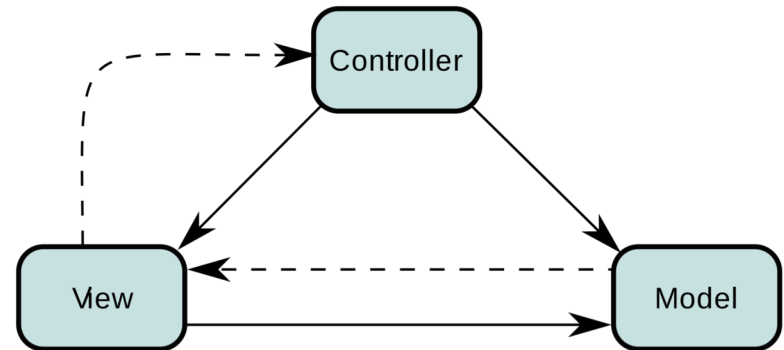
# Camada de Apresentação

- Arquitectura
  - Como organizar a camada de interface?
    - Construção da componente gráfica
    - Resposta aos eventos
    - Comunicação com a lógica de negócio



# Camada de Apresentação

- Model - View - Controller *pattern*



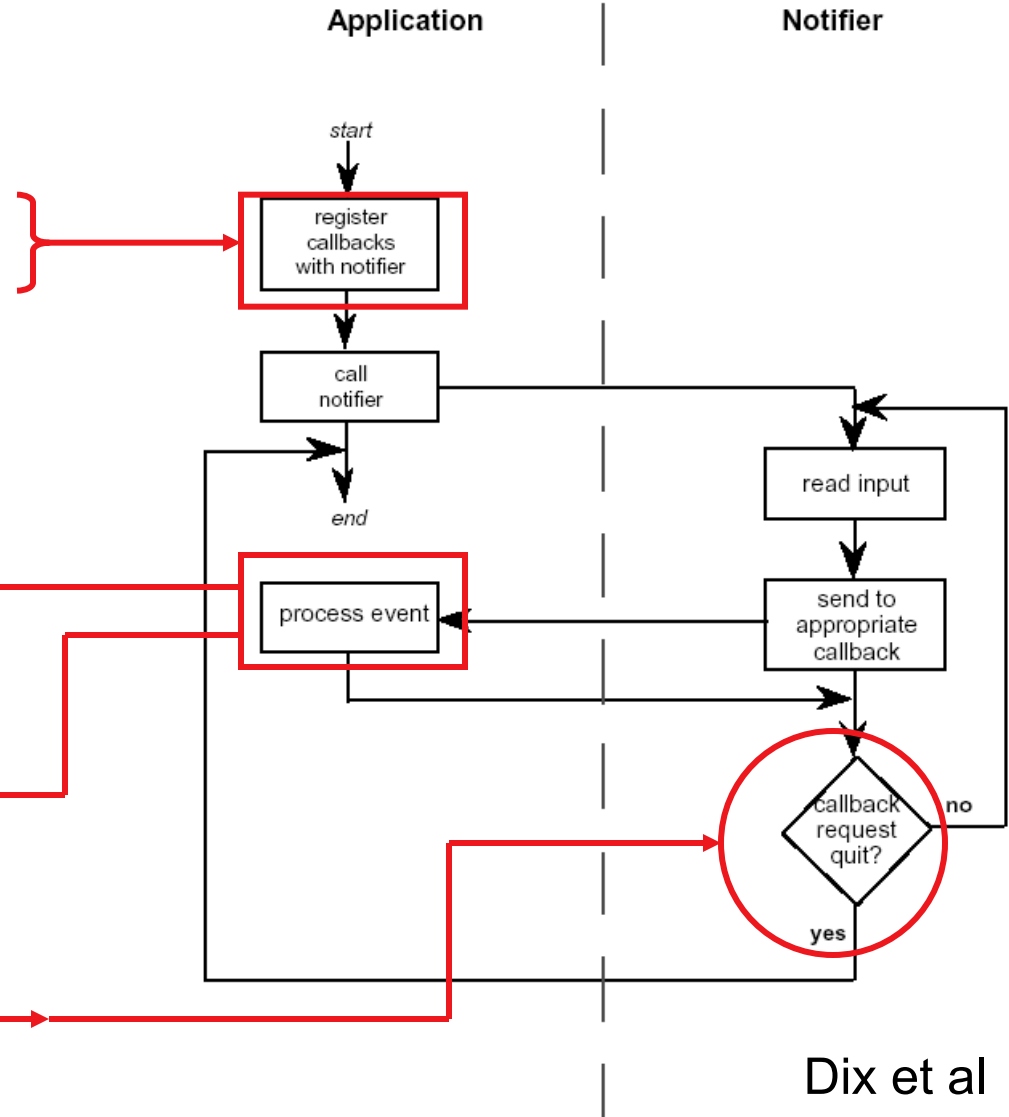
- Model: Business Logic
  - core program functionality and data
- View: A apresentação dos dados da aplicação. Desenho da interface e seus componentes
- Controller: Processa e responde aos eventos
  - Handles user input
  - Translates interface events into program functions

# Event Handling

```
void main(String[] args) {  
    Menu menu = new Menu();  
    menu.setOption("Save");  
    menu.setOption("Quit");  
    menu.setAction("Save", mySave)  
    menu.setAction("Quit", myQuit)  
    ...  
}
```

```
int mySave(Event e) {  
    // save the current file  
}
```

```
int myQuit(Event e) {  
    // close down  
}
```





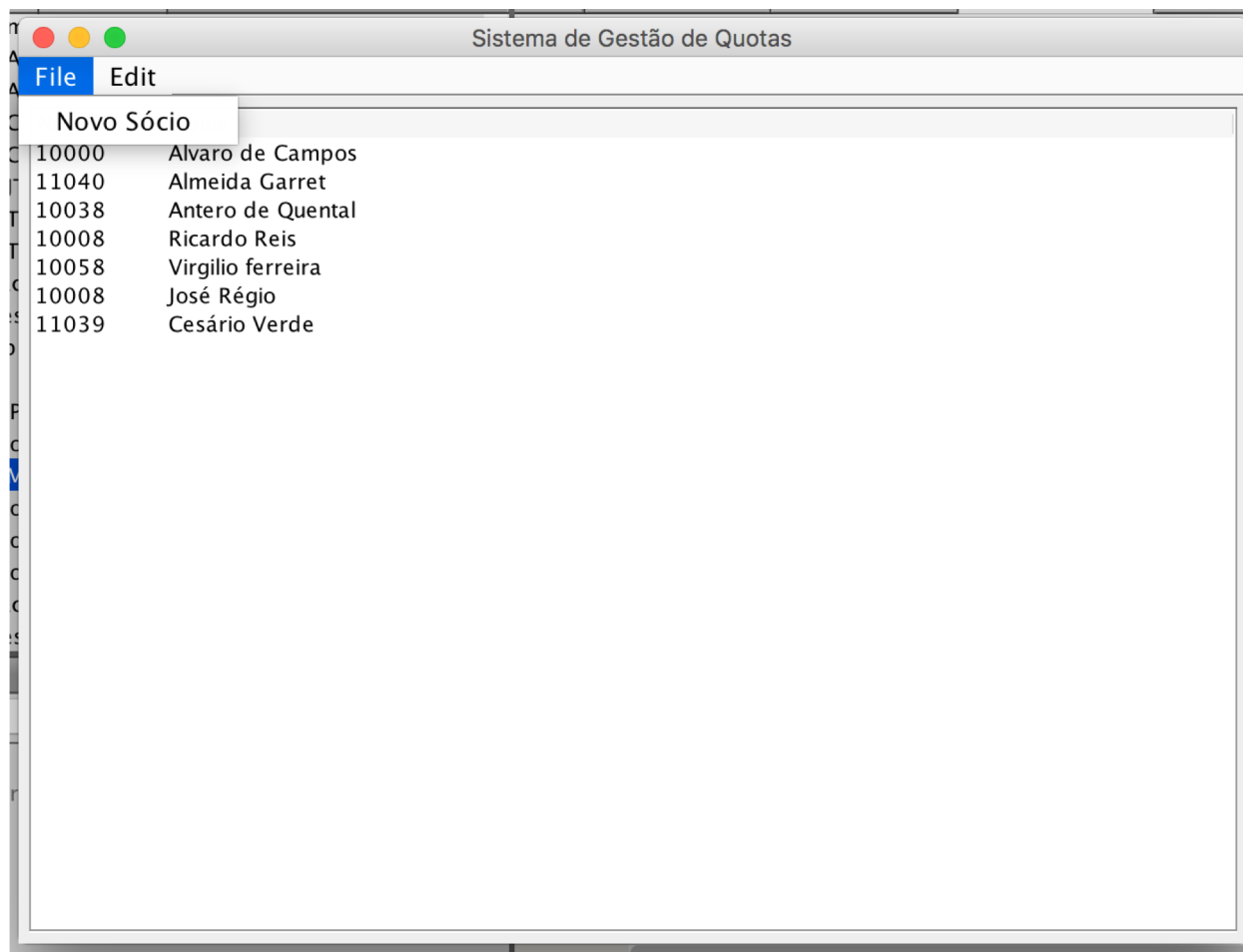
# Event handling

- low-level events
  - window system events
  - component (hierarchy) events
  - low-level input
- semantic events
  - action events
  - items events
  - may correspond to a set of low-level events (may differ over look and feels)
- listen to semantic rather than low-level events where possible

# Projecto sobre gestão de sócios

- Fazer um software que permita gerir a lista de sócios, e respectivas quotas, de uma organização estilo Cesium.
- Métodos da camada de negócio:
  - `public Map<Integer, Aluno> getAlunos();`
  - `public void pagarQuota(Integer numero, Double valor)`
  - `public Aluno getAluno(int num)`
  - `public void addAluno(Aluno a)`
- Podem acrescentar métodos que julguem necessários
- De seguida apresentam-se algumas interfaces exemplo de uma (possível) resolução do problema.

# Visualizar os sócios



# Inserir novo sócio

The screenshot shows a Java Swing application titled "Sistema de Gestão de Quotas". The main window has a menu bar with "File" and "Edit". It contains a table with the following data:

| Número | Nome              |
|--------|-------------------|
| 10000  | Álvaro de Campos  |
| 11040  | Almeida Garret    |
| 10038  | Antero de Quental |
| 10008  | Ricardo Reis      |
| 10058  | Virgílio ferreira |
| 10008  | José Régio        |
| 11039  | Cesário Verde     |

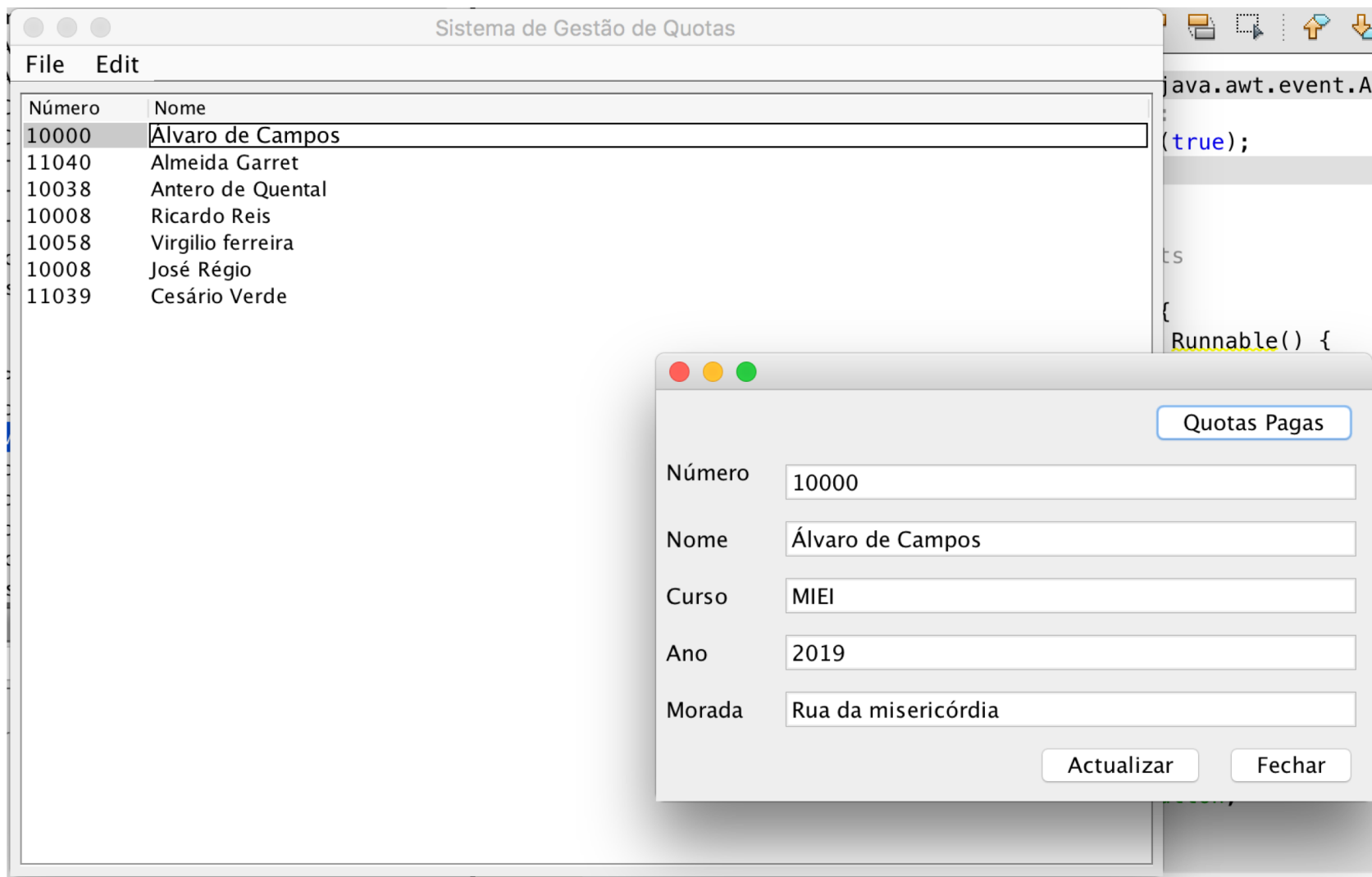
Overlaid on the main window is a modal dialog box for adding a new member. It has the following fields and buttons:

- Número**: A text input field.
- Nome**: A text input field.
- Curso**: A text input field.
- Ano**: A text input field.
- Morada**: A text input field.
- Buttons**: "Guardar" (Save) and "Fechar" (Close).

In the background, a portion of a Java code editor is visible, showing the following code:

```
java.awt.event.Ac  
(true);  
  
ts  
{  
    Runnable() {
```

# Detalhes de um sócio



# Visualizar as quotas de um sócio

The screenshot displays a Java Swing application titled "Sistema de Gestão de Quotas". The main window contains a table with two columns: "Número" and "Nome". The table lists three shareholders: 10000 (Álvaro de Campos), 11040 (Almeida Garret), and 10038 (Antero de Quental). A modal dialog titled "Quotas Pagas" is open, showing a table of payments for the selected shareholder (10000). The table has two columns: "Mes" and "Valor". The payments are for January, February, and March 2019, each with a value of 5.0. The modal also includes a "Pagar Quotas" button and an "OK" button. In the background, another modal is visible with fields for "Quotas Pagas", "10000", "Álvaro de Campos", "MIEI", "2019", and "Rua da misericórdia", along with "Actualizar" and "Fechar" buttons. A code editor on the right shows Java code related to the application.

| Número | Nome              |
|--------|-------------------|
| 10000  | Álvaro de Campos  |
| 11040  | Almeida Garret    |
| 10038  | Antero de Quental |

| Mes        | Valor |
|------------|-------|
| 2019-01-01 | 5.0   |
| 2019-02-01 | 5.0   |
| 2019-03-01 | 5.0   |