ENGENHARIA DE SOFTWARE

PROJETO DE ARQUITETURA

Data: 22/10/2024

PROF. MARCIO NAGY

AGENDA

- Objetivos de Projeto de Arquitetura
- Requisitos não funcionais
- Padrão MVC (Modelo-Visão-Controlador)
- Exercício 01 - Descreva e desenhe o fluxo da aplicação de um usuário acessando um site de notícias utilizando diagrama UML.
- Exercício 02 - Sistema de Gerenciamento de Biblioteca

Objetivos de Projeto de Arquitetura

- Visa compreender como um sistema de software deve ser organizado e projetar a estrutura geral desse sistema.
- Gera um modelo que descreve como o sistema está organizado como um conjunto de componentes que se comunicam.
- Nos processos ágeis desenvolve um estágio inicial que esteja focado na criação do projeto da arquitetura geral do sistema.

Projeto de Arquitetura

- Vantagens de projetar e documentar explicitamente uma arquitetura de software:
  - Comunicação de stakeholders – apresentação em alto nível para discussão com diferentes stakeholders
  - Análise de Sistema – Explicita decisões de arquitetura para atender requisitos não funcionais.
  - Reuso em larga escala – Arquitetura de sistema é muitas vezes a mesma para sistemas com requisitos similares.

REQUISITOS NÃO FUNCIONAIS

DESEMPENHO

QUAL O CRITÉRIO DE DESEMPENHO QUE O SISTEMA DEVERÁ ATENDER? O CRITÉRIO É MENSURÁVEL?

REQUISITO NÃO FUNCIONAL

DISPONIBILIDADE

QUAL O DE DISPONIBILIDADE E ACESSIBILIDADE O SISTEMA DEVERÁ ATENDER? É POSSÍVEL VERIFICAR ESTE CRITÉRIO

REQUISITO NÃO FUNCIONAL

SEGURANÇA

QUAIS AS CONDIÇÕES DE SEGURANÇA QUE O SISTEMA DEVE GARANTIR? COMO É POSSÍVEL MEDIR ESTAS CONDIÇÕES?

REQUISITO NÃO FUNCIONAL

INTEGRABILIDADE

QUAIS MÉTODOS DE INTEGRAÇÃO E INTEROPERABILIDADE O SISTEMA DEVE ATENDER? COMO TESTAR ESTES MÉTODOS?

REQUISITO NÃO FUNCIONAL

MANUTENIBILIDADE

Qual a criticidade da manutenibilidade, usando componentes pequenos e autocontidos que possam ser facilmente modificados ?

REQUISITO NÃO FUNCIONAL

Projeto de Arquitetura - MVC (Modelo-Visão-Controlador)

- O padrão MVC é um padrão de arquitetura de software amplamente utilizado para organizar e estruturar aplicações, separando as responsabilidades em três componentes interconectados.
- Este padrão promove a modularidade, a reusabilidade de código e a manutenibilidade, tornando-se uma escolha popular para desenvolvimento web e de aplicações complexas.

Padrão MVC (Modelo-Visão-Controlador)

- Separa a apresentação e a interação dos dados do sistema.
- Utilizado quando há várias maneiras de visualizar e interagir com os dados e quando os requisitos futuros para interação e apresentação dos dados são desconhecidos.
- Estruturado em três componentes lógicos que interagem entre si:

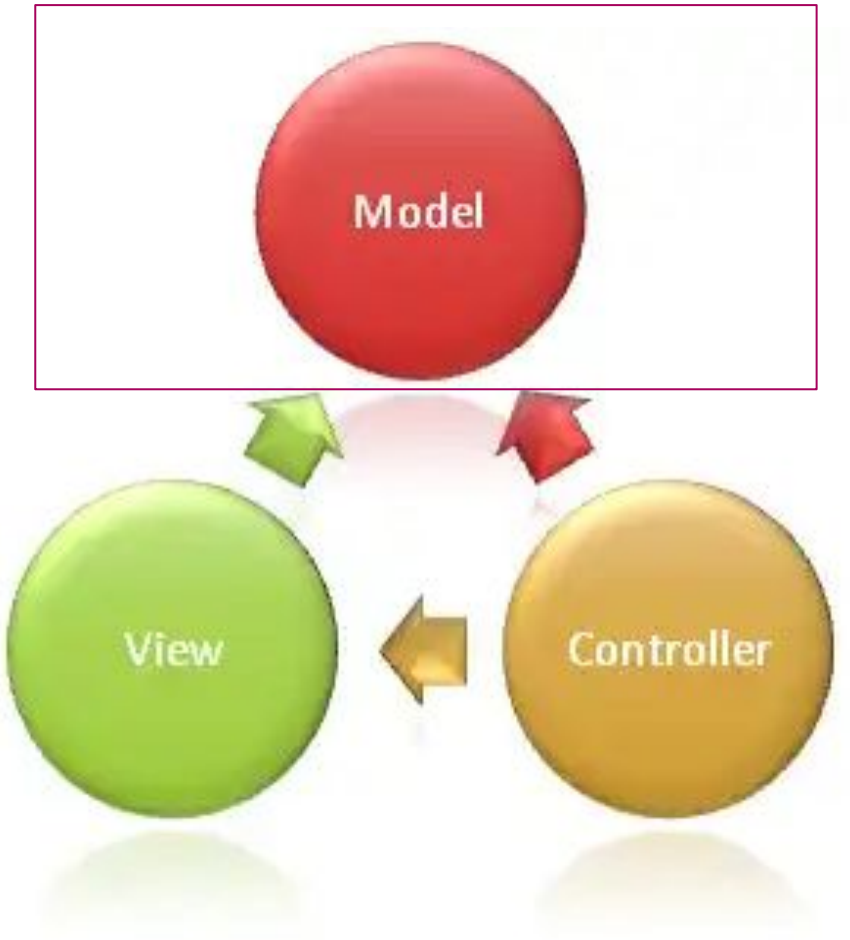
Padrão MVC (Modelo-Visão-Controlador)

➔ **Modelo:** Representa os dados da aplicação e a lógica de negócio. Gerencia o estado, a persistência e a manipulação dos dados.

➔ **Visão:** Responsável por apresentar os dados ao usuário. Exibe as informações do modelo de forma organizada e interativa.

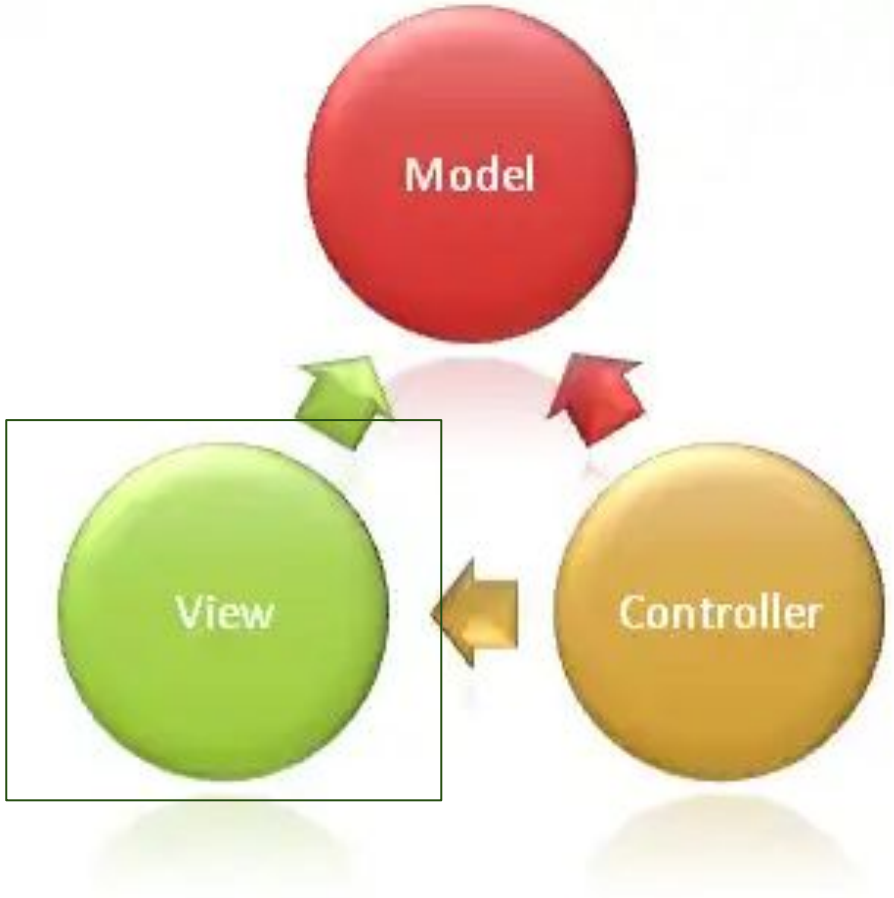
➔ **Controlador:** Faz a intermediação entre o modelo e a visão. Recebe as ações do usuário, processa-as e atualiza o modelo ou a visão conforme necessário.

Padrão MVC (Camada Modelo)



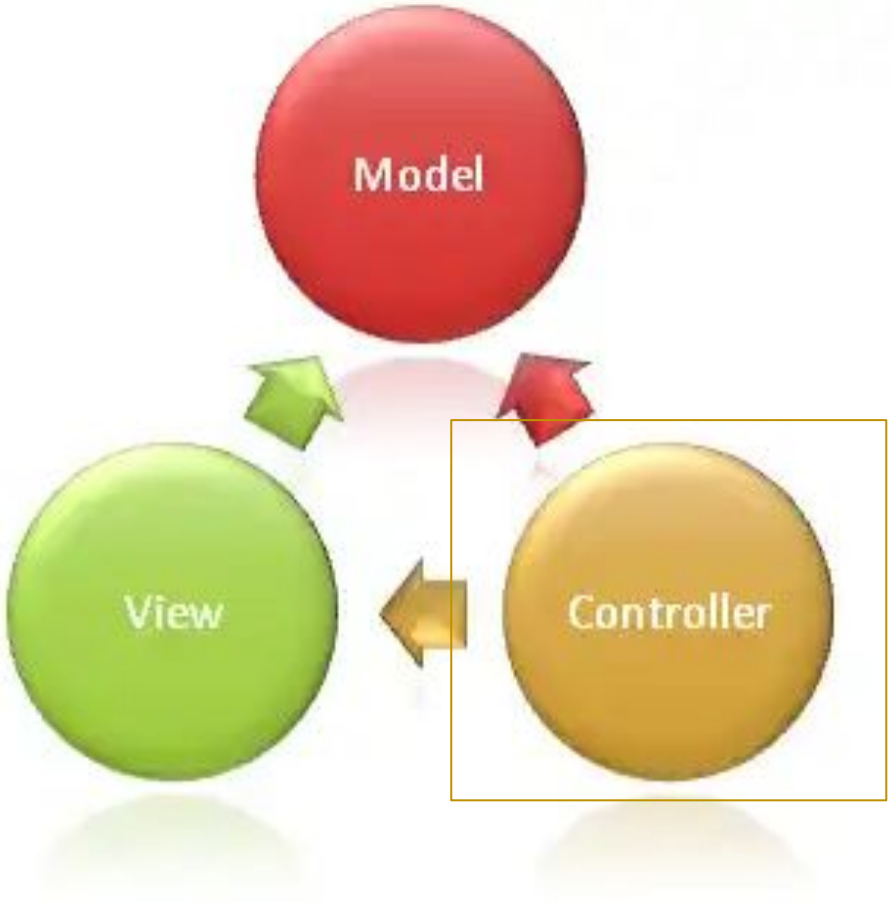
- **Responsabilidades:**
  - Representação dos dados da aplicação (entidades, objetos, etc.).
  - Regras de negócio e lógica da aplicação.
  - Validação de dados.
  - Persistência de dados (armazenamento e recuperação).
  - Notificação de mudanças no estado dos dados para a visão.
- Exemplos: Classes, interfaces, estruturas de dados, acesso a banco de dados.

Padrão MVC (Camada Visão)



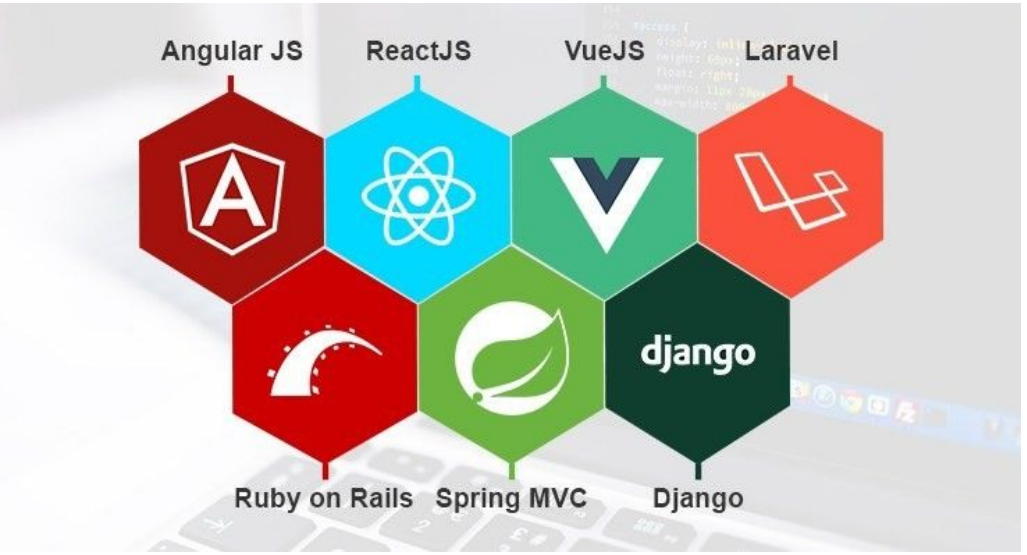
- **Responsabilidades:**
  - Apresentação dos dados ao usuário.
  - Interface gráfica com o usuário (GUI).
  - Formatação e organização dos dados para exibição.
  - Recebimento de entrada do usuário.
  - Não contém lógica de negócio complexa.
- Exemplos: páginas HTML, templates, componentes de interface, layouts.

Padrão MVC - Camada Controlador



- **Responsabilidades:**
  - Receber e processar requisições do usuário.
  - Interagir com o modelo para acessar ou modificar dados.
  - Selecionar a visão apropriada para exibir os resultados.
  - Gerenciar o fluxo de navegação da aplicação.
  - Não contém lógica de negócio complexa, delega para o modelo.
- Exemplos: Servlets, Actions, Controllers em frameworks web.

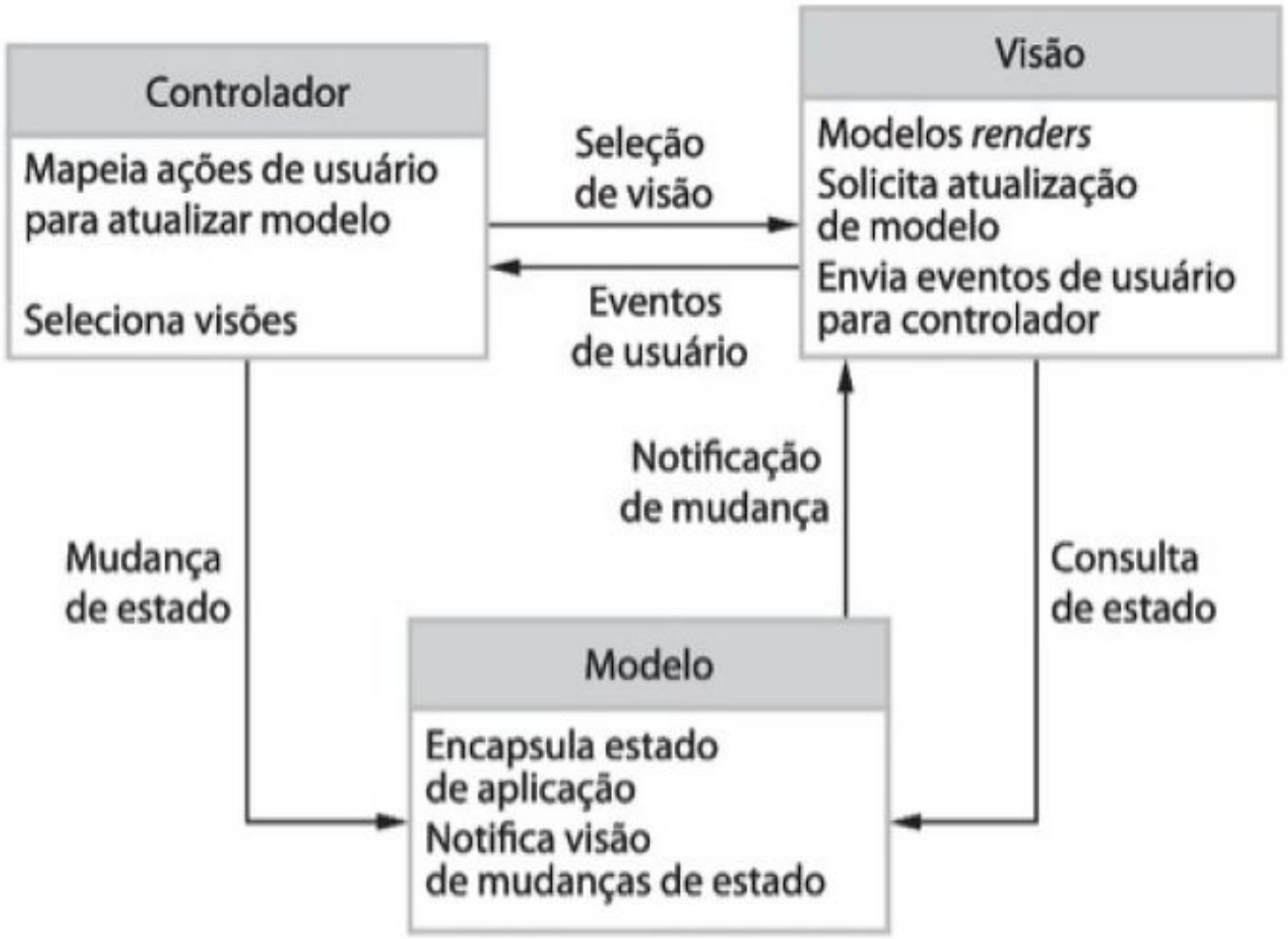
Padrão MVC - Frameworks



- Diversos frameworks web populares utilizam o padrão MVC:
  - **Java:** Spring MVC, Struts
  - **Python:** Django, Flask
  - **PHP:** Laravel, CodeIgniter
  - **Ruby:** Ruby on Rails
  - **JavaScript:** Angular, React (com adaptações)

- Frameworks MVC fornecem estruturas, bibliotecas e ferramentas que facilitam o desenvolvimento de aplicações web seguindo o padrão.

Visão conceitual do padrão MVC



Benefícios do MVC:

- **Separação de responsabilidades:** Cada componente tem uma função específica, tornando o código mais organizado, fácil de entender e manter.
- **Reutilização de código:** Os componentes podem ser reutilizados em diferentes partes da aplicação.
- **Facilidade de teste:** Os componentes podem ser testados individualmente.
- **Desenvolvimento paralelo:** Diferentes desenvolvedores podem trabalhar em diferentes componentes simultaneamente

MVC em aplicação web

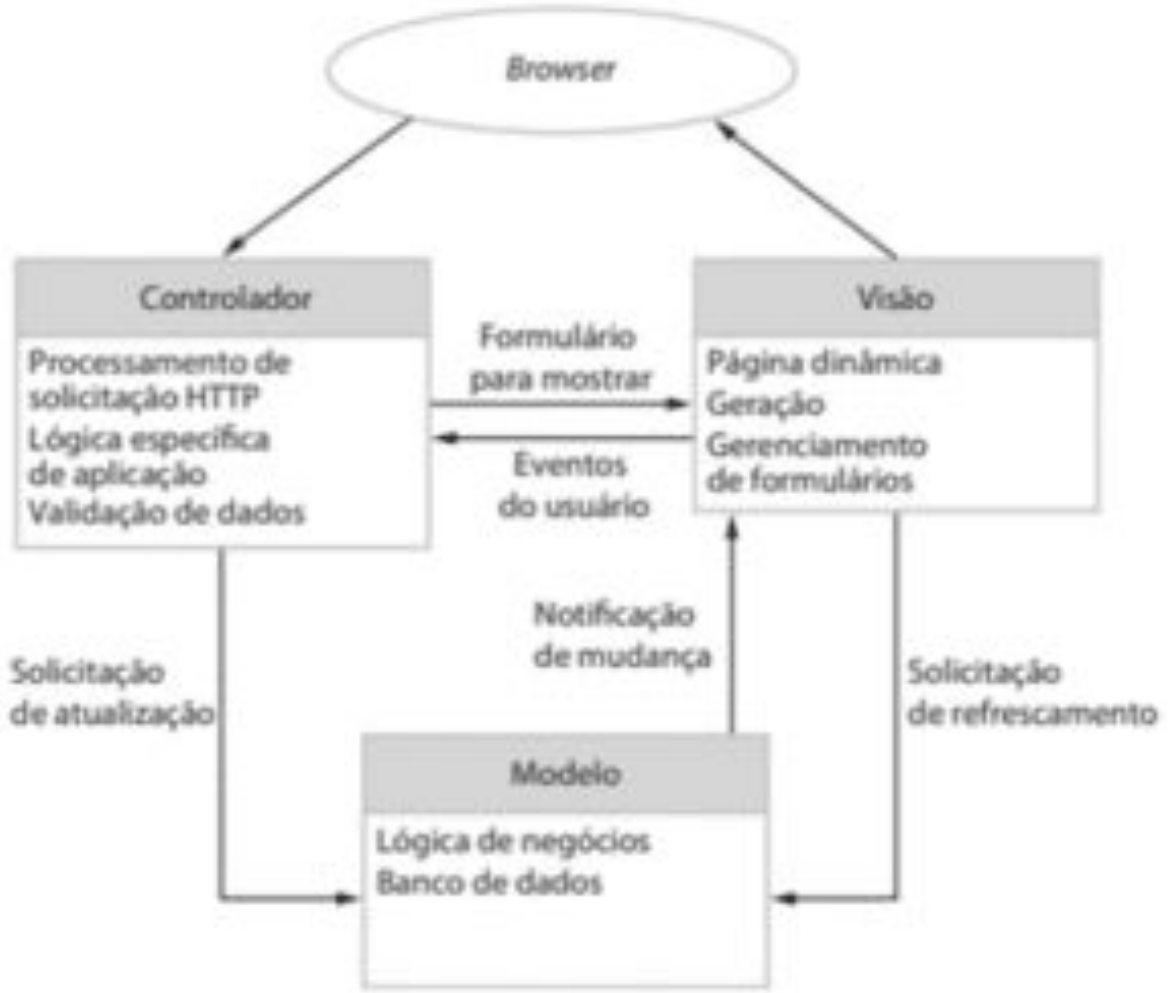
Descrição do Diagrama:

- **Usuário:** Inicia a interação com a aplicação web através do navegador.
- **Navegador:** Envia requisições para o controlador.
- **Controlador:** Recebe as requisições do navegador, interage com o modelo para processar os dados e seleciona a view apropriada para exibir a resposta.
- **Modelo:** Contém a lógica de negócio da aplicação, os dados e as regras de acesso a dados.
- **Visão:** Apresenta as informações ao usuário.

MVC em aplicação web

Fluxo da Aplicação:

1. O usuário interage com a aplicação web através do **navegador**, enviando uma requisição (ex: clicar em um link, enviar um formulário).
2. O **controlador** recebe a requisição e a interpreta.
3. O **controlador** interage com o **modelo** para processar a requisição, buscar ou atualizar dados.
4. O **modelo** acessa o banco de dados (se necessário) e retorna os dados para o **controlador**.
5. O **controlador** seleciona a **view** apropriada para exibir o resultado.
6. A **view** recebe os dados do **controlador** e gera o HTML para ser exibido no **navegador**.
7. O **navegador** recebe o HTML e o exibe para o usuário.



Exercício 01 - Descreva e desenhe o fluxo da aplicação de um usuário acessando um site de notícias.

1) Descreva o fluxo de eventos da aplicação.
2) Descreva o Diagrama de sequência UML mostrando o fluxo de eventos desta aplicação.
3) Utilize o plantuml.com ou PlantText para gerar a imagem do Diagrama UML.

Resolução Exercício 01 - descreva e desenhe o fluxo da aplicação de um usuário acessando um site de notícias.

1) **Descreva o fluxo de eventos.**
   1. O usuário clica no link da notícia no navegador.
   2. O navegador envia uma requisição HTTP GET para o controlador, incluindo o ID da notícia no link.
   3. O controlador recebe a requisição e solicita ao modelo a notícia correspondente ao ID.
   4. O modelo acessa o banco de dados para buscar os dados da notícia.
   5. O banco de dados retorna os dados da notícia para o modelo.
   6. O modelo formata os dados e os retorna para o controlador.
   7. O controlador envia os dados da notícia para a view.
   8. A view renderiza o HTML da notícia utilizando os dados recebidos.
   9. A view retorna o HTML renderizado para o controlador.
   10. O controlador envia o HTML para o navegador.
   11. O navegador exibe a notícia formatada para o usuário.

.

Resolução Exercício 01 - descreva e desenhe o fluxo da aplicação de um usuário acessando um site de notícias.

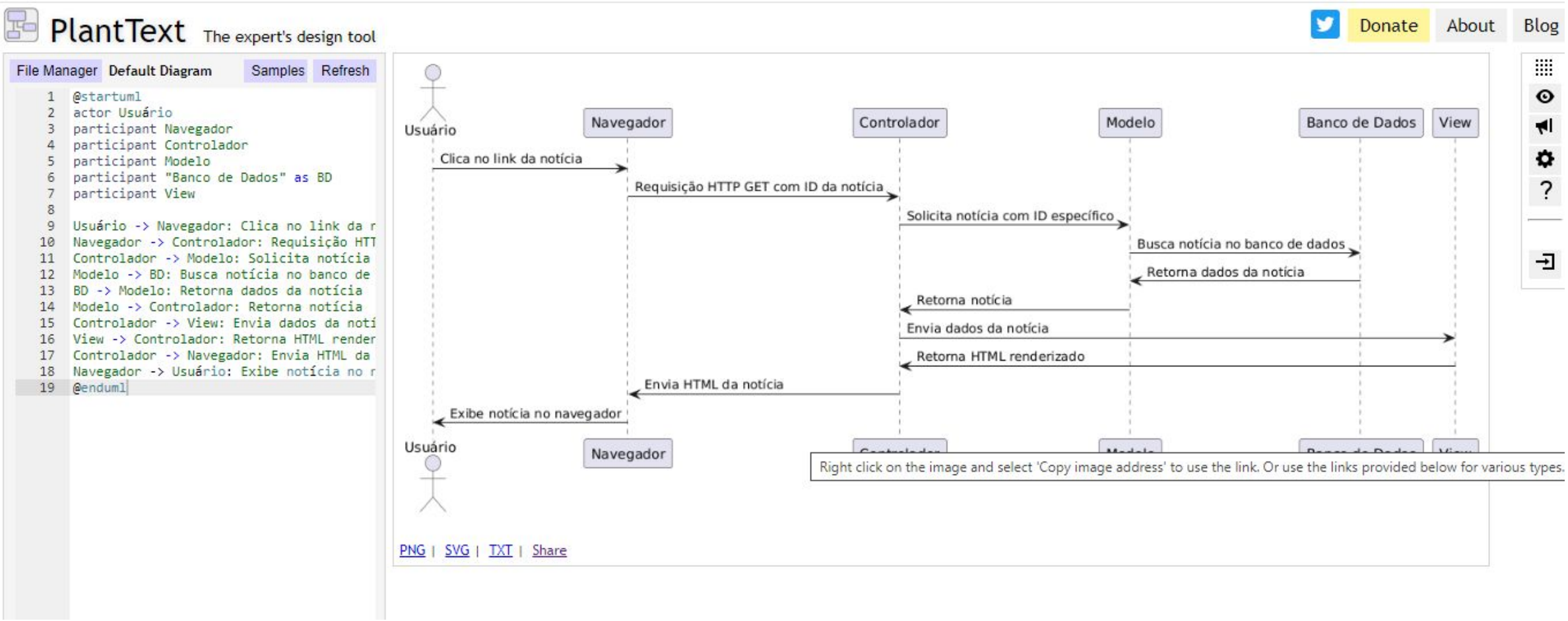
2) Descreva o Diagrama de sequência UML mostrando o fluxo de eventos desta aplicação

```
@startuml
actor Usuário
participant Navegador
participant Controlador
participant Modelo
participant "Banco de Dados" as BD
participant View

Usuário -> Navegador: Clica no link da notícia
Navegador -> Controlador: Requisição HTTP GET com ID da notícia
Controlador -> Modelo: Solicita notícia com ID específico
Modelo -> BD: Busca notícia no banco de dados
BD -> Modelo: Retorna dados da notícia
Modelo -> Controlador: Retorna notícia
Controlador -> View: Envia dados da notícia
View -> Controlador: Retorna HTML renderizado
Controlador -> Navegador: Envia HTML da notícia
Navegador -> Usuário: Exibe notícia no navegador
@enduml
```

Resolução Exercício 01 - descreva e desenhe o fluxo da aplicação de um usuário acessando um site de notícias.

3) Utilize o plantuml.com ou PlantText para gerar a imagem do Diagrama UML.



Exercício 02 - Sistema de Gerenciamento de Biblioteca.

1) Desenvolver um sistema web para gerenciar uma biblioteca, com as seguintes funcionalidades:
    - Cadastro de livros (título, autor, ISBN).
    - Consulta de livros por título ou autor.
    - Empréstimo de livros para usuários cadastrados.
    - Devolução de livros.

1) Utilize o padrão MVC para estruturar a aplicação, com utilização de conceitos básicos de orientação a objetos e sem a utilização de frameworks web específicos.
2) Defina as classes do modelo, visão e controlador.
3) Descreva o fluxo de eventos da aplicação.
4) Utilize PlantUML ou PlantText para gerar o fluxo e a imagem do Diagrama UML
5) Implemente as funcionalidades básicas em Python ou outra linguagem que desejar (cadastro, consulta, empréstimo e devolução).
6) (Desafio Opcional) Utilize um framework web MVC de sua preferência.
7) Entrega para o dia 28/10/2024 - Grupo de até 03 pessoas.

Obrigado!