UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ

QXD0252 - Engenharia de Software - 2024.1
Modelos e Arquiteturas de Software

Grupo : Andressa Lima Colares, Carlos Ryan Santos Silva, Davi José Lima de Sousa.

- **Tipo de arquitetura**

Para esta aplicação web, um site com viés informativo, que publica artigos sobre história do Brasil, será utilizada a arquitetura MVC (Model-View-Controller).

1. **Modelo -** Nesta camada, será apresentada as classes Usuario, Artigo e Comentário, pois essas serão as classes que lidam com a manipulação de dados e, caso necessário, dados recuperados.Para seu desenvolvimento, será utilizado JavaScript (Node js).
2. **Visão -** É a parte da qual o usuário mais terá contato na aplicação, onde irá ser exibido as informações de forma simples para fácil entendimento. Será desenvolvida em JavaScript
3. **Controlador -**, utilizando React como framework. É o intermediário entre Modelo e Visão, pois é ele que recebe a solicitação do usuário. Para essa aplicação, o Controlador receberá as requisições do protocolo HTTP, que chamará os métodos apropriados do Modelo e irá selecionar a os métodos da Visão que irá exibir a requisição solicitada pelo usuário.

Diagrama de atividades - Este diagrama irá representar o fluxo de atividades e interações que ocorrerão nesta aplicação .

- **Principais atividades :**

1. **Usuário leitor -** lê artigos, favoritar artigos, comentar em publicações, enviar artigos para revisão.
2. **Administrador -** Revisa comentários escritos por usuários, remove comentários desnecessários, restringe usuários, envia artigos para o usuário revisor, corrige e remove artigos.
3. **Usuário Revisor -** corrige e revisa os artigos publicados, enviados pelo administrador, publica os artigos no web site.

- **Fluxo de atividades (Diagrama do Usuário)**

- **Ler artigo -** O usuário leitor acessa e lê os artigos disponíveis na aplicação.
- **Favoritar artigo -** Se o usuário leitor desejar ler um artigo posteriormente, ele tem a opção de favoritá-lo para leitura posterior.
- **Comentar em publicações -** O usuário leitor pode comentar nos artigos que ele leu.

- **Enviar artigos para revisão -** Se o usuário leitor desejar publicar um artigo , ele poderá enviar para revisão.

- **Fluxo de atividades (Diagrama do Administrador)**
- **Revisar comentários -** O administrador verifica os comentários postados pelos usuários.
- **Excluir comentários -** Comentários inadequados são removidos.
- **Restringir usuário -** O administrador poderá dar punições por comportamentos inadequados.
- **Corrigir Artigos:** Se necessário, o administrador corrige os artigos antes de enviá-los para revisão.
- **Enviar Artigo para Usuário Revisor:** Os artigos são enviados para o usuário revisor para revisão adicional.
- **Remove artigos**: Artigos com erros e não revisados poderão ser removidos pelo administrador.

- **Fluxo de atividades (Diagrama Usuário Revisor)**
- **Corrigir e Revisar Artigos:** O usuário revisor corrige e revisa os artigos enviados pelo administrador.
- **Enviar Artigo Revisado para Publicação:** Após a revisão, o usuário revisor envia os artigos revisados de volta para o sistema para publicação.

Diagrama de classes - O principal objetivo desse diagrama de classes é representar a interação entre os diferentes tipos de usuário que a aplicação possuirá.

Classes da aplicação:

1. UsuárioModel:

a) **Atributos:**
id: int
usuario: string
senha: string
tipo: string
favoritos : ArtigoModel()

2. **ArtigoModel:**

a) **Atributos:**
id: int
titulo: string
conteudo: string
dataPublicação: date
comentarios: ComentarioModel()

3. ComentarioModel:

a) **Atributos**

texto: string
id: int
dataHora: dateTime
usuarioid : int

4. RevisorController:

a) **Métodos:**

revisarArtigo(artigo_id: int): void

5. LeitorController:

a) **Métodos**:

salvarArtigo (artigo: ArtigoModel):void
comentarArtigo (comentario: ComentarioModel, artigo : ArtigoModel) : void
enviarArtigo (artigoModel) : void

6. AdministradorController:

a) **Métodos:**

revisarComentario(comentario_id: int): void
removeArtigo (artigo_id:int):void
corrigirArtigo(artigo_id: int, novoConteudo: string): void
removerComentario(comentario_id: int): void
mandarParaRevisor(artigo_id: int): void
restringeUsuario (Usuario:UsuarioModel):vo

7. LeitorView:

a) **Métodos:**

lerArtigo (artigo: ArtigoModel): void
atualizar():void
buscarArtigo(titulo:string):ArtigoModel

8. Administrador View:

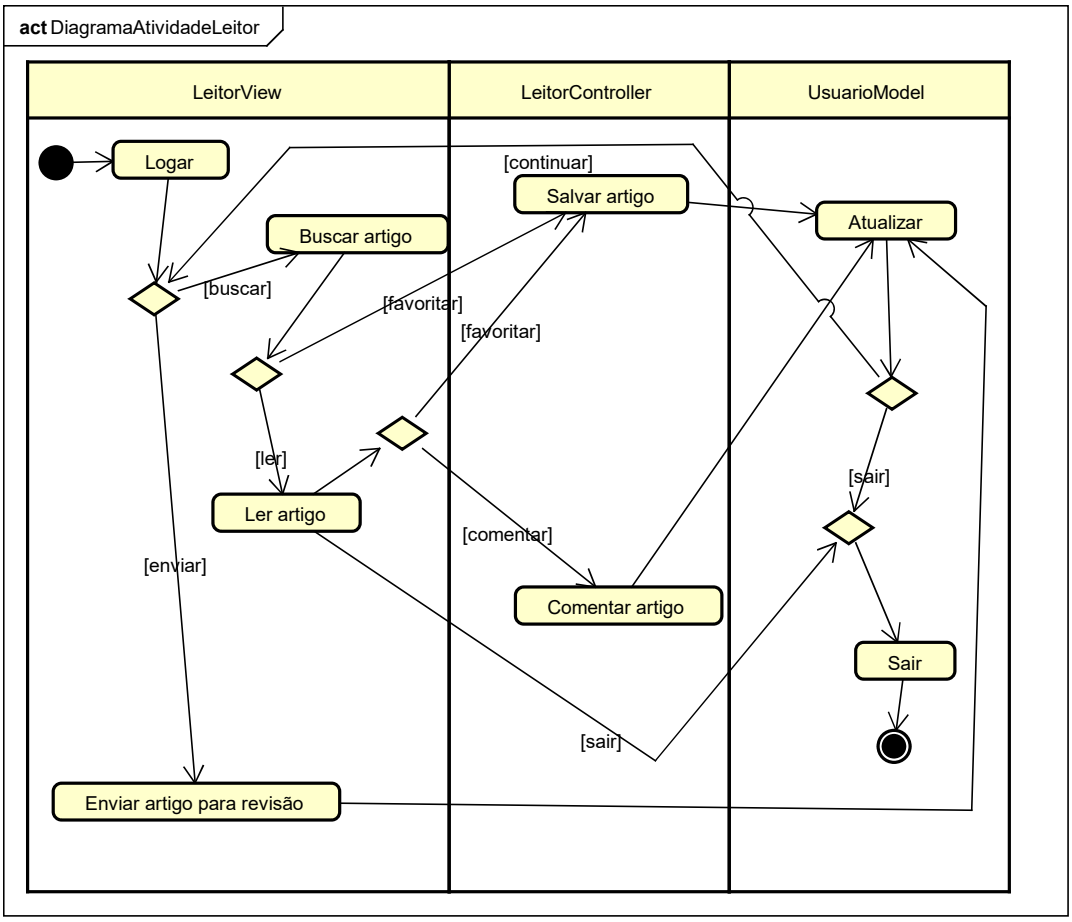
a) **Métodos:**

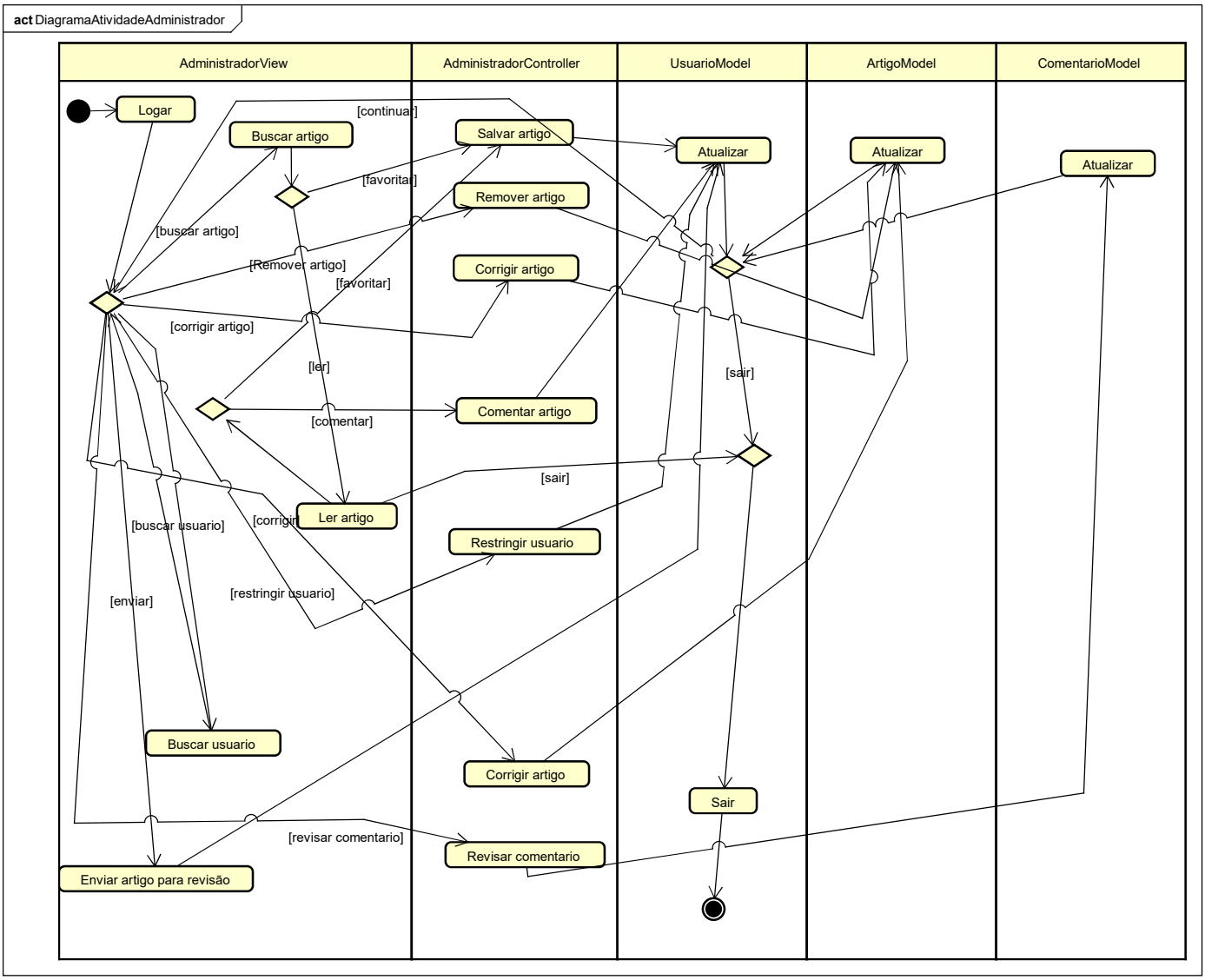
obterUsuario(usuario: string): UsuarioModel

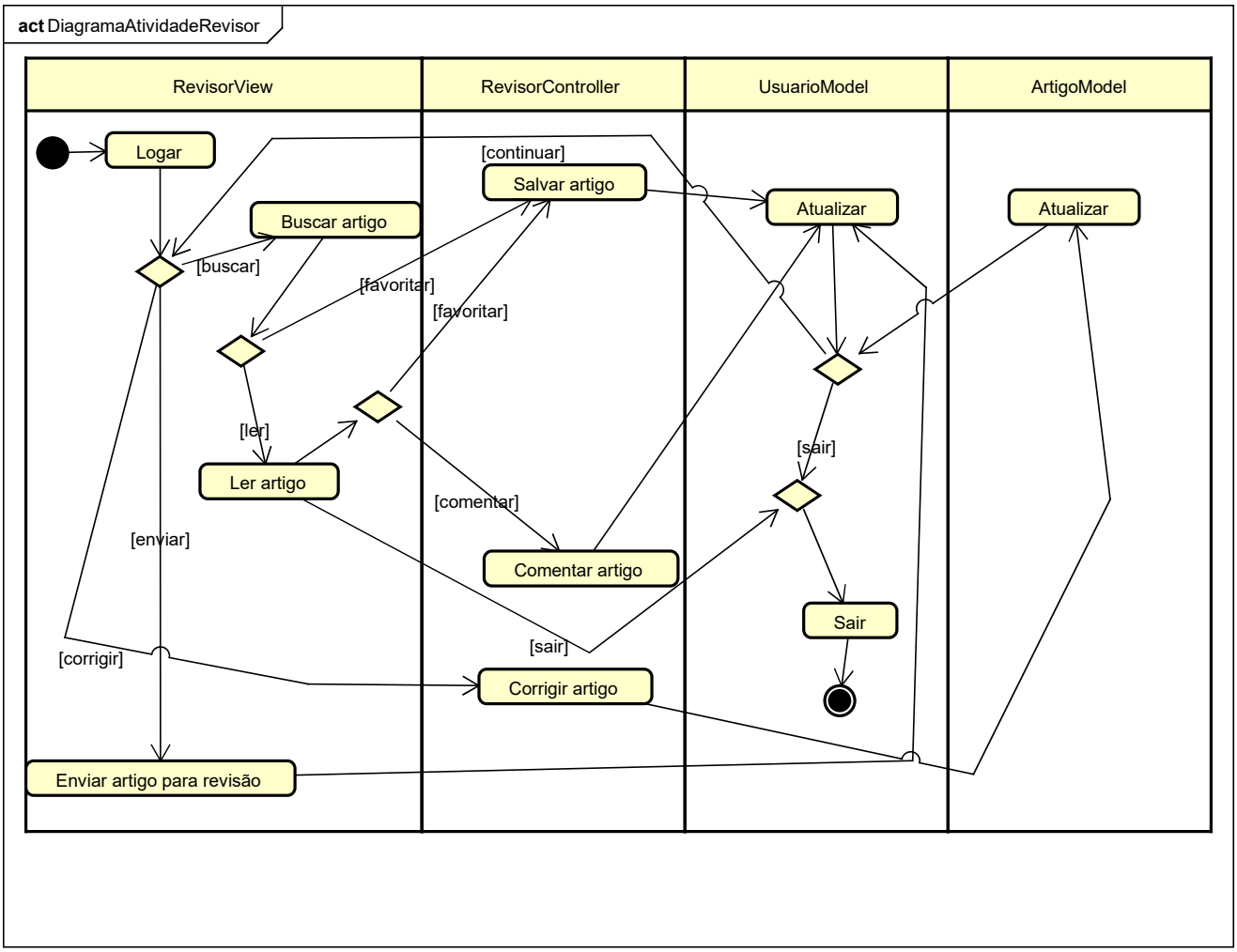
- **Descrição das classes:**

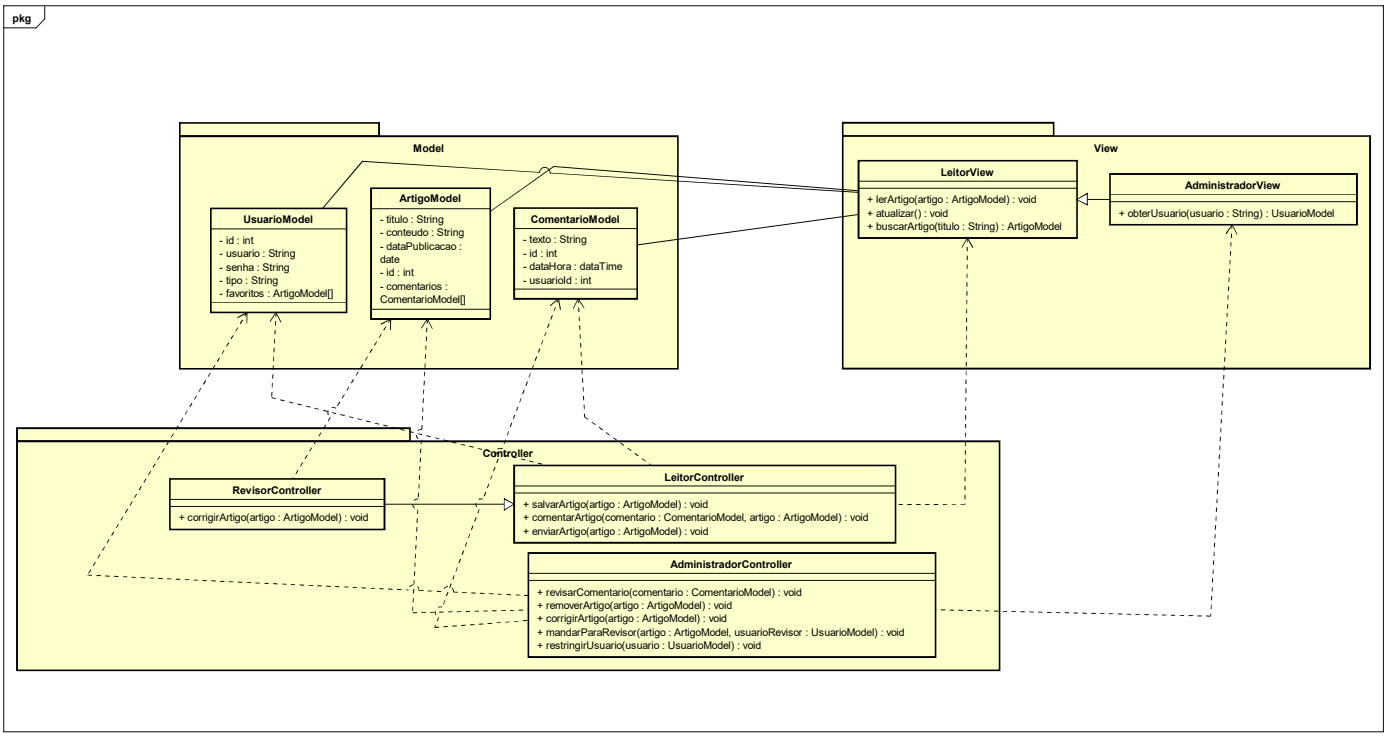
- **UsuarioModel :** Representa um usuário genérico da aplicação, podendo ele ser um usuário comum, administrador ou revisor.Possui atributos básicos como nome, usuário, senha e id.
- **AritgoModel:** Classe que representa um artigo publicado nesta aplicação.
- **ComentarioModel**: Representa os comentários publicados pelo usuário, ele possui id próprio, possui também data da publicação e ligação com usuário que o publicou.

- **RevisorController:** Ele poderá revisar os artigos enviados para publicação, corrigir ou vetar possíveis erros.
- **LeitorController**: Poderá salvar ou comentar nos artigos publicados e também poderá enviar artigos para o usuário Revisor.
- **AdministradorController**: Este usuário poderá revisar comentários, remover artigos por possíveis erros, corrigir erros em publicações, remover comentários indesejados de usuários, enviar artigos para o revisor, e punir usuários por comentários ofensivos ou impróprios.
- **LeitorView:** Poderá ler os artigos publicados, atualizar o feed da página inicial da aplicação e buscar um artigo que deseja.
- **AdministradorView**: Poderá buscar um usuário na aplicação.


act DiagramaAtividadeLeitor
LeitorView
LeitorController
UsuarioModel
Logar
[continuar]
Salvar artigo
Atualizar
Buscar artigo
[buscar]
[favoritar]
[favoritar]
[ler]
[sair]
Ler artigo
[comentar]
[enviar]
Comentar artigo
Sair
[sair]
Enviar artigo para revisão


act DiagramaAtividadeAdministrador
AdministradorView
AdministradorController
UsuarioModel
ArtigoModel
ComentarioModel
Logar
Buscar artigo
Salvar artigo
Atualizar
Atualizar
Atualizar
[continua]
[favoritar]
Remover artigo
[buscar artigo]
[Remover artigo]
[favoritar]
Corrigir artigo
[corrigir artigo]
[ler]
[sair]
[comentar]
Comentar artigo
[sair]
[buscar usuario]
[corrigir]
Ler artigo
Restringir usuario
[enviar]
[restringir usuario]
Buscar usuario
Corrigir artigo
Sair
[revisar comentario]
Revisar comentario
Enviar artigo para revisão


act DiagramaAtividadeRevisor
RevisorView
RevisorController
UsuarioModel
ArtigoModel
Logar
[continuar]
Salvar artigo
Atualizar
Atualizar
Buscar artigo
[buscar]
[favoritar]
[favoritar]
[ler]
Ler artigo
[sair]
[comentar]
[enviar]
Comentar artigo
Sair
[sair]
[corrigir]
Corrigir artigo
Enviar artigo para revisão


pkg
Model
UsuarioModel
- id : int
- usuario : String
- senha : String
- tipo : String
- favoritos : ArtigoModel[]
ArtigoModel
- titulo : String
- conteudo : String
- dataPublicacao : date
- id : int
- comentarios : ComentarioModel[]
ComentarioModel
- texto : String
- id : int
- dataHora : dataTime
- usuarioId : int
View
LeitorView
+ lerArtigo(artigo : ArtigoModel) : void
+ atualizar() : void
+ buscarArtigo(titulo : String) : ArtigoModel
AdministradorView
+ obterUsuario(usuario : String) : UsuarioModel
Controller
RevisorController
+ corrigirArtigo(artigo : ArtigoModel) : void
LeitorController
+ salvarArtigo(artigo : ArtigoModel) : void
+ comentarArtigo(comentario : ComentarioModel, artigo : ArtigoModel) : void
+ enviarArtigo(artigo : ArtigoModel) : void
AdministradorController
+ revisarComentario(comentario : ComentarioModel) : void
+ removerArtigo(artigo : ArtigoModel) : void
+ corrigirArtigo(artigo : ArtigoModel) : void
+ mandarParaRevisor(artigo : ArtigoModel, usuarioRevisor : UsuarioModel) : void
+ restringirUsuario(usuario : UsuarioModel) : void