DOCUMENTAÇÃO TÉCNICA DE SISTEMA: ONCRED

A **OnCred** é uma plataforma web de fórum de discussões estruturada com foco na mitigação da desinformação em ambientes virtuais por meio da exibição transparente de credenciais e usabilidade inclusiva.

1. Visão Geral da Arquitetura

O sistema foi desenvolvido utilizando uma arquitetura monolítica organizada sob o padrão de projeto **MVC (Model-View-Controller)**.

📦 Forum

├ 📂 Controller

| ├ HomeController.php

| ├ PerfilController.php

| ├ PostController.php

| └ UsuarioController.php

├ 📂 DAO

| ├ HomeDAO.php

| ├ PerfilDAO.php

| ├ PostDAO.php

| └ UsuarioDAO.php

├ 📂 generic

| ├ Acao.php

| ├ Autoload.php

| ├ Conexao.php

| ├ Controller.php

```
|  └ MysqlFactory.php
├ 📂 public
|  ├ CriarPost.php
|  ├ Home.php
|  ├ Perfil.php
|  └ Post.php
├ 📂 Service
|  ├ HomeService.php
|  ├ PerfilService.php
|  ├ PostService.php
|  └ UsuarioService.php
├ 📂 Style
|  ├ criarPost.css
|  ├ home2.css
|  ├ post.css
|  └ oncredLogo.svg
├ 📂 template
|  ├ HomeTemplate.php
|  ├ ITemplate.php
|  ├ PerfilTemplate.php
|  ├ PostTemplate.php
|  └ UsuarioTemplate.php
└ .htaccess
└ index.php
```

2. Tecnologias Empregadas

- **Front-End:** HTML5 Semântico, CSS3 e JavaScript.
- **Back-End:** PHP.
- **Banco de Dados:** MySQL.

3. Mapeamento do Banco de Dados

`usuarios`

- `id` (INT, PK, AutoIncrement, Tam:16)
- `nome` (VARCHAR, Tam:40)
- `usuario` (VARCHAR, Unique, Tam:40)
- `email` (VARCHAR, Unique, Tam:40)
- `senha` (VARCHAR, Tam:40)
- `biografia` (VARCHAR, Nullable, Tam:500)

`posts`

- `post_id` (INT, PK, AutoIncrement, Tam:11)
- `usuario_id` (INT, FK -> `usuarios.id`, Tam:16)
- `titulo` (VARCHAR, Tam:200)
- `conteudo` (VARCHAR, Tam:1000)
- `categoria` (VARCHAR, Tam:40)
- `tags` (VARCHAR, Tam:40)
- `data` (TIMESTAMP)

`comentarios`

- `comentario_id` (INT, PK, AutoIncrement, Tam:16)
- `post_id` (INT, FK -> `posts.id`, Tam:11)
- `usuario_id` (INT, FK -> `usuarios.id`, Tam:16)
- `conteudo` (VARCHAR, Tam:500)
- `data_criacao` (TIMESTAMP)

`credenciais`

- `credencial_id` (INT, PK, AutoIncrement, Tam:11)
- `usuario_id` (INT, FK -> `usuarios.id`, Tam:16)
- `nome` (VARCHAR, Tam:50)
- `descricao` (TEXT, Tam:200)

`interacoes`

- `interacao_id` (INT, PK, AutoIncrement, Tam:11)
- `usuario_id` (INT, FK -> `usuarios.id`, Tam:16)
- `post_id` (INT, FK -> `posts.id`, Tam:11)
- `tipo` (INT, Tam:11)

4. Fluxo de Execução

4.1. Fluxos de Gestão de Usuário (Usuario)

A. Registrar Novo Usuário (registrar)

1. O método recebe os dados do formulário (nome, usuario, email, senha) via $_POST.
2. Instancia UsuarioService e invoca o método registrar(...) repassando os parâmetros para persistência.
3. Define explicitamente a flag de estado global $_SESSION["logado"] = true.
4. Redireciona o navegador para a rota de home via cabeçalho HTTP.

B. Autenticação de Usuário (login)

1. Captura as credenciais informadas (email, senha) via $_POST.
2. Invoca UsuarioService->login($email, $senha), que retorna uma matriz com os dados do usuário se as credenciais forem válidas.
3. Inicia a sessão e avalia se o retorno contém registros (count($usuario) > 0):
    - **Sucesso:** Alimenta a $_SESSION com o nome de usuário (usuario), o ID (id) e define logado como true. Redireciona para home.
    - **Falha:** Destrói a sessão ativa (session_destroy()) e redireciona para home#erro injetando uma flag de erro na URL.

C. Atualizar Dados Cadastrais (atualizarConta)

1. Recebe email, nova_senha e usuario_id do formulário postado.
2. **Processamento Condicional:**
    - Se o campo email não estiver vazio (!empty), aciona UsuarioService->atualizarContaEmail(...).
    - Se o campo senha não estiver vazio, aciona UsuarioService->atualizarContaSenha(...).
3. Redireciona o usuário de volta para sua página de perfil (perfil?id=...).

D. Encerramento de Conta (deletarConta)

1. Lê o identificador único do usuário via parâmetro de URL $_GET['usuario_id'].
2. Delega para UsuarioService->deletarConta($id) para que remova o registro e vínculos relacionados do MySQL.
3. Redireciona para a rota limpa para realizar a purga final da sessão.

2. Fluxos de Tópicos e Interações (Post)

A. Renderização da Página do Tópico (Post)

1. Garante a inicialização da sessão (session_start()) se não estiver ativa.
2. Através do ID recebido via $_GET['id'], realiza disparos síncronos para a PostService para coletar:
    - Dados do artigo (getPost), Comentários (getComentarios), Usuários participantes (getIDUsuarioComentarios) e Mapeamento de Votos/Interações (getInteracoes).
3. Varre o array de usuários comentadores e monta um mapa estruturado indexando as credenciais e competências técnicas individuais de cada autor de comentário (getCredenciaisPorUsuario).
4. Percorre o post para extrair e definir numericamente a quantidade total de comentários atrelados (getCountComentarios).
5. Invoca o motor de gabarito técnico $this->template->layout(...) passando o arquivo Post.php e injetando todo o array de dados consolidados.

B. Publicação de Novo Conteúdo (criarPost)

1. Inicializa a sessão e recupera o ID do autor ativo ($_SESSION['id_usuario']).
2. Coleta titulo, categoria, tags e o bloco de conteudo via $_POST.
3. Dispara a criação técnica no banco via PostService->criarPost(...).
4. Redireciona o fluxo da aplicação para a página inicial (home).

C. Inserção de Comentários (criarComentario)

1. Coleta a identidade do usuário da sessão atual.
2. Captura post_id e o texto do conteudo. Caso o conteúdo esteja vazio, aborta a inserção imediatamente e redireciona à tela do post apontando a falha na URL (#erroComentario).
3. Se válido, grava via PostService->criarComentario(...).
4. Recarrega a página atualizando a árvore de discussões (post?id=...).

D. Processamento de Votos e Favoritos (upvote / downvote / salvar)

Os três métodos seguem uma engenharia lógica de validação de estado de interação (Tipo 1: Upvote, Tipo 2: Downvote, Tipo 3: Salvo):

1. O controlador consulta a Service para averiguar se o usuário já possui interações salvas naquela publicação.
2. **Máquina de Estados:**
    - Caso o usuário clique em *Upvote*, mas possua um voto do tipo *Downvote* ativo, o sistema altera o registro diretamente para Upvote (mudarInteracao) e retorna com a âncora #mudarUp.
    - Caso clique em *Upvote* e ele já tenha feito essa mesma ação anteriormente, o sistema interpreta como remoção e exclui o voto (deletarInteracao), retornando com #deletarUp.

    - Caso não haja registros prévios, computa uma nova inserção direta do voto ou salvamento (upvote / downvote / salvar).
3. Redireciona de forma imediata para o link do post correspondente mantendo o foco do usuário.

3. Fluxos de Perfil Técnico (PerfilController)

A. Renderização e Consolidação Unificada do Perfil (`Perfil`)

1. Inicializa o contexto de sessão e recupera o identificador do perfil alvo (`$_GET['id']`).
2. Realiza chamadas síncronas sequenciais à `PerfilControllerService` para preencher as coleções base:
    - Coleção total de posts favoritados/salvos (`getPostsSalvos`).
    - Coleção total de posts publicados pelo usuário (`getPostsID`).
    - Histórico de comentários realizados (`getComentarios`).
    - Dados demográficos do perfil (`getUsuarioDados`), passando o sinalizador de tipo `0`.
    - Matriz de qualificações e certificações validadas (`getCredenciais`).
3. Recupera a contagem agregada de comentários do usuário (`getCountComentariosUsuario`), tratando valores nulos com operador de coalescência (`?? 0`).
    - Recupera a contagem de posts publicados pelo usuário (`getCountPosts`).
4. Percorre iterativamente cada post criado pelo usuário para extrair e montar a matriz `$contagens_por_posts`, mapeando dinamicamente o total de comentários, *upvotes* e *downvotes* de cada um.
5. Executa um segundo laço estruturado focado nas publicações favoritadas, construindo de forma isolada a matriz `$contagens_por_posts_salvos` (armazenando comentários e balanceamento de votos de terceiros).
6. Percorre o array `$comentarios` associando dinamicamente o ID de cada postagem ao seu respectivo título textual (`getPostsPostID`), garantindo a rastreabilidade e clareza visual do histórico na interface.
7. Itera sobre os posts salvos para identificar e mapear o nome do autor original de cada publicação favoritada, disparando `getUsuarioDados` com o sinalizador de tipo `1`.
8. Consolida todas as variáveis, arrays estruturados e matrizes de contagem e os injeta na renderização do arquivo `Perfil.php` via `PerfilControllerTemp`.

B. Gestão de Biografia e Credenciais de Confiabilidade

- Captura a alteração textual enviada pelo formulário através do campo `$_POST['biographyEdit']`, obtém o identificador do autor diretamente

da sessão ativa (`$_SESSION['id_usuario']`), delega a persistência no MySQL à camada de serviço e força o recarregamento do perfil atualizado.

- Intercepta o título da competência técnica institucional (`credentialName`) e o memorial de validação correspondente (`credentialDesc`) inseridos pelo usuário, gravando o novo registro associado ao ID da sessão antes de redirecionar o navegador.
- Captura os identificadores únicos tanto da credencial (`$_GET['id_credencial']`) quanto do usuário alvo (`$_GET['id_usuario']`) diretamente por parâmetros de URL HTTP GET, aciona a purga do vínculo na base de dados e devolve o fluxo à tela de perfil aplicando a âncora de interface `#del`.

4. Fluxo da Interface Principal (HomeController)

A. Listagem Geral e Filtragem Temática (Home)

1. Abre o escopo de sessão. Caso a flag $_SESSION['logado'] falhe ou não exista, o controlador limpa o array global de sessão, destrói os cookies identificadores (session_destroy) e força o deslogamento automático do usuário.
2. Solicita os posts gerais (getPosts) e a árvore lateral de categorias de fóruns cadastrados (getCategories).
3. Realiza um laço de repetição estruturado extraindo individualmente, post por post, o total de comentários e interações ativas para exibição precisa nos excertos da tela principal.
4. Avalia a existência de requisições de filtragem via $_GET['categoria']:
    - **Sim:** Substitui o array de postagens padrão chamando getPostsByCategory($categoria) e injeta o parâmetro de seleção ativa na renderização da View Home.php.
    - **Não:** Renderiza a View Home.php em seu estado bruto original contendo toda a listagem global cronológica do sistema.

B. Purga Completa de Sessão (Limpa)

1. Inicializa o contexto, zera o array global com $_SESSION = [] e liquida o token de identificação ativo no servidor por meio do comando session_destroy().
2. Envia o usuário deslogado e limpo de volta para a raiz do index da home.