

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\main.py

```
# main.py
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from core.configs import settings
from utils.scheduler import tarefa_diaria, scheduler, tarefa_limpar_relatorios
import uvicorn
from contextlib import asynccontextmanager
from api.v1.endpoints.categoria import router as categoria_router
from api.v1.endpoints.setor import router as setor_router
from api.v1.endpoints.usuario import router as usuario_router
from api.v1.endpoints.item import router as item_router
from api.v1.endpoints.retirada import router as retirada_router
from api.v1.endpoints.relatorios import router as relatorio_router
from api.v1.endpoints.alerta import router as alerta_router
from fastapi.staticfiles import StaticFiles
from frontend.routes.home import router as frontend_router
import mimetypes

# Importar o roteador WebSocket e o manager
from utils.websocket_endpoints import websocket_router, manager

# Forçar o tipo MIME para arquivos .js. Isso deve ser feito ANTES de StaticFiles ser montado.
mimetypes.add_type('application/javascript', '.js')
mimetypes.add_type('application/javascript', '.mjs') # Para módulos ES6 com extensão .mjs

@asynccontextmanager
async def lifespan(app: FastAPI):
    # Executado antes do app iniciar
    try:
        scheduler.add_job(tarefa_diaria, 'cron', hour=8, minute=1) # verificar validade dos produtos
        scheduler.add_job(tarefa_limpar_relatorios, 'cron', hour=11, minute=8) # limpar relatórios todos os dias
        scheduler.start()
        print("Scheduler iniciado com sucesso via lifespan.")
    except Exception as e:
        print("Erro ao iniciar scheduler via lifespan:", e)
    yield # app roda
    # Executado quando o app estiver encerrando
    scheduler.shutdown()
    print("Scheduler finalizado.")

app = FastAPI(
    title="Sistema de Gerenciamento de Almoxarifado",
    description="API para gerenciar o estoque e retirada de materiais",
    version="1.0.0",
    lifespan=lifespan
)

app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:8082"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Incluindo os endpoints Back-end
app.include_router(setor_router, prefix=settings.API_STR, tags=['Gerenciamento de Setores'])
app.include_router(categoria_router, prefix=settings.API_STR, tags=['Gerenciamento de Categorias'])
app.include_router(usuario_router, prefix=settings.API_STR, tags=['Gerenciamento de Usuários'])
```

```

app.include_router(item_router, prefix=settings.API_STR, tags=['Gerenciamento de Itens'])
app.include_router(retirada_router, prefix=settings.API_STR, tags=['Gerenciamento de Retiradas'])
app.include_router(relatorio_router, prefix=settings.API_STR, tags=['Geração de Relatórios de Itens'])
app.include_router(alerta_router, prefix=settings.API_STR, tags=['Gerenciamento de Alertas'])

# Incluir o roteador WebSocket
# Adicionado um parâmetro para o user_id no WebSocket, que será opcional na rota
app.include_router(websocket_router, prefix=settings.API_STR) # Prefixo para o WebSocket

# Montar pasta de arquivos estáticos
app.mount("/static", StaticFiles(directory="frontend/static"), name="static")

# Incluindo os endpoints Front-End
app.include_router(frontend_router)

if __name__ == "__main__":
    uvicorn.run("main:app", host="0.0.0.0", port=8082, reload=True)

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\routes\home.py

```

#frontend\routes\home.py

from fastapi import APIRouter, Request, Cookie
from fastapi.responses import HTMLResponse, RedirectResponse
from fastapi.templating import Jinja2Templates
from jwt import decode, PyJWTError
from core.configs import settings

router = APIRouter()
templates = Jinja2Templates(directory="frontend/templates")

@router.get("/", response_class=HTMLResponse)
async def home(request: Request, access_token: str | None = Cookie(None)):
    print("access_token recebido:", access_token)

    if access_token:
        try:
            payload = decode(
                access_token,
                settings.JWT_SECRET,
                algorithms=[settings.ALGORITHM]
            )
            tipo = payload.get("tipo_usuario")
            if tipo == 1:
                return RedirectResponse("/dashboardServidor", status_code=303)
            elif tipo == 2:
                return RedirectResponse("/dashboardAlmoxarifado", status_code=303)
            elif tipo == 3:
                return RedirectResponse("/dashboardDirecao", status_code=303)
        except PyJWTError:
            pass # Cookie inválido ou expirado

    return templates.TemplateResponse("index.html", {"request": request})

@router.get("/dashboardAlmoxarifado", response_class=HTMLResponse)
async def dashboard_almoxarifado(request: Request):
    return templates.TemplateResponse("dashboardAlmoxarifado.html", {"request": request})

@router.get("/dashboardServidor", response_class=HTMLResponse)

```

```

async def dashboard_servidor(request: Request):
    return templates.TemplateResponse("dashboardServidor.html", {"request": request})

@router.get("/dashboardDirecao", response_class=HTMLResponse)
async def dashboard_direcao(request: Request):
    return templates.TemplateResponse("dashboardDirecao.html", {"request": request})

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\css\bootstrap.min.css

```

@charset "UTF-8";/*!
 * Bootstrap v5.3.5 (https://getbootstrap.com/)
 * Copyright 2011-2025 The Bootstrap Authors
 * Licensed under MIT (https://github.com/twbs/bootstrap/blob/main/LICENSE)
 */:root,[data-bs-theme=light]{--bs-blue:#0d6efd;--bs-indigo:#6610f2;--bs-purple:#6f42c1;--bs-pink:#d63384;--bs-red:#dc3545;--bs-orange:#fd7e14;--bs-yellow:#ffc107;--bs-green:#28a745;--bs-teal:#20a997;--bs-cyan:#17a2b8;--bs-black:#343a40;--bs-white:#fff;--bs-light:#f8f9fa;--bs-dark:#343a40;--bs-primary:#007bff;--bs-secondary:#6c757d;--bs-success:#28a745;--bs-info:#17a2b8;--bs-warning:#ffc107;--bs-danger:#dc3545;--bs-light:#f8f9fa;--bs-dark:#343a40}
/*# sourceMappingURL=bootstrap.min.css.map */

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\css\custom.css

```

/* frontend\static\css\custom.css */

/* === Header superior === */

.custom-header {
    background-color: #0052cc;
    color: #fff;
}

.custom-header header-logo {
    height: 40px;
}

.custom-header .header-title {
    font-weight: bold;
    font-size: 1rem;
}

/* === Navbar === */

.custom-navbar {
    background-color: #0052cc;
}

/* Dropdown menus */

.navbar-nav .dropdown-menu {
    border-radius: 0.25rem;
    box-shadow: 0 0.5rem 1rem rgba(0,0,0,0.1);
}

/* Itens do dropdown */

.navbar-nav .dropdown-item { /* Corrigido de .navbar-nav.dropdown-item */
    padding: 0.5rem 1rem;
    font-size: 0.95rem;
}

```

```

.navbar-nav .dropdown-item:hover { /* Corrigido de .navbar-nav.dropdown-item:hover */
  background-color: #f8f9fa;
}

/* Links do navbar */

.nav-link {
  padding: 0.5rem 1rem;
}

.nav-link:hover, .nav-link:focus {
  background-color: rgba(255,255,255,0.1);
  border-radius: 0.25rem;
}

/* Ícones */

.navbar-nav .bi {
  font-size: 1.3rem;
  margin: 0 0.3rem;
  cursor: pointer;
}

/* Ajuste responsivo: evitar que o ícone de dropdown quebre linha */

.nav-item.dropdown {
  white-space: nowrap;
}

/* Estilos para o contêiner de toasts (alvo a classe Bootstrap diretamente) */
.toast-container {
  top: 1rem !important; /* Ajusta a distância do topo */
  right: 1rem !important; /* Ajusta a distância da direita */
  max-width: 350px !important; /* Define uma largura máxima para o contêiner de toasts */
  width: 90% !important; /* Garante que em telas menores, ele ocupe 90% da largura */
  padding: 0 !important; /* Remove padding duplicado se já houver no p-3 */
  margin: 0 !important;
}

.toast {
  width: 100% !important;
  word-wrap: break-word !important;
  box-shadow: 0 0.5rem 1rem rgba(0, 0, 0, 0.15) !important;
  margin-bottom: 0.5rem !important; /* Espaçamento entre toasts empilhados */
}

/* Ajusta o corpo do toast para não ter padding excessivo que o corte */
.toast-body {
  padding-right: 1.5rem !important; /* Garante espaço para o botão de fechar */
}

/* Estilos para os itens do dashboard de relatórios */

.report-item-style {
  background-color: #f8f9fa; /* Um tom claro para o fundo */
  border-radius: 0.5rem; /* Bordas arredondadas */
  margin-bottom: 0.5rem; /* Espaçamento entre os itens */
  transition: all 0.2s ease-in-out; /* Transição suave para hover */
  border: 1px solid #e9ecef; /* Uma borda sutil */
}

.report-item-style:hover {
  background-color: #e2e6ea; /* Um tom um pouco mais escuro ao passar o mouse */
}

```

```

        transform: translateY(-2px); /* Efeito de "levantar" */
        box-shadow: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.075); /* Sombra sutil */
        border-color: #dee2e6;
    }

    .report-item-style i {
        margin-left: 1rem; /* Espaçamento entre o texto e o ícone */
    }

    /* Estilos para o sino de notificação */
    .notification-dot {
        width: 12px;
        height: 12px;
        border-radius: 50%;
        background-color: red;
        display: block;
        position: absolute;

        /* Posicionamento base, será modificado pela animação */
        top: 6px;
        right: 12px;

        z-index: 1070;
        animation: pulse 1.5s infinite; /* animação */
    }

    /* Animação de pulso */
    @keyframes pulse {
        0% {
            opacity: 1;
        }
        50% {
            opacity: 0.6;
        }
        100% {
            opacity: 1;
        }
    }

    .nav-link.position-relative {
        position: relative;
        padding-right: 15px; /*para dar espaço à bolinha no lado direito */
        padding-top: 5px; /* para dar espaço à bolinha no lado superior */
    }

    /* Garante que ambos os botões (Editar/Deletar) tenham a mesma largura */
    .btn-acoes {
        min-width: 5rem; /* ou qualquer valor que você julgar ideal */
        text-align: center; /* centraliza o ícone/texto dentro do botão */
    }

    /* Em telas muito estreitas, eles se empilham (graças ao flex-wrap).
       Para espaçamento vertical maior, garantimos margem inferior quando empilhados. */
    @media (max-width: 576px) {
        .btn-acoes {
            width: 100%; /* ocupam toda a largura da célula */
            margin-bottom: 0.5rem; /* 0.5rem de espaçamento entre eles empilhados */
        }
    }

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\css\dashboards.css

```
/* static/css/dashboards.css */

/* Cores principais */
:root {
  --blue-top:    #0052cc;
  --blue-nav:    #003166;
}

/* Barra superior */
.top-bar {
  background-color: var(--blue-top);
  color: #fff;
  padding: 0.5rem 1rem;
}

.top-bar .logo {
  font-size: 1.25rem;
  font-weight: 600;
  text-align: center;
  color: white;
}

/* Navbar secundária */
.navbar-custom {
  background-color: var(--blue-nav);
}

.navbar-custom .nav-link,
.navbar-custom .navbar-brand,
.navbar-custom .btn-link {
  color: #fff;
}

.navbar-custom .nav-link:hover,
.navbar-custom .btn-link:hover {
  color: #ddd;
}

/* Ajuste para centralizar os menus dropdown */
.navbar-nav {
  display: flex;
  justify-content: center;
  width: 100%;
}

.nav-item.dropdown {
  flex: 0 1 auto; /* Não cresce, pode encolher, largura automática */
  white-space: nowrap; /* Impede que o texto quebre em várias linhas */
  padding: 0 10px; /* Espaçamento entre os itens */
}

.dropdown-menu {
  min-width: 12rem;
}

/* Estilos para os cards de acesso rápido */
.quick-access-card {
  padding: 1.5rem;
  background-color: white;
}
```

```

border-radius: 0.375rem;
box-shadow: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.075);
text-align: center;
transition: transform 0.2s;
height: 100%;
}

.quick-access-card:hover {
  transform: translateY(-5px);
}

.quick-access-card i {
  font-size: 2rem;
  color: #0d6efd;
}

.quick-access-card p {
  margin-top: 0.75rem;
  margin-bottom: 0;
}

/* Ajuste para mobile */
@media (max-width: 992px) {
  .navbar-nav {
    flex-direction: column;
  }

  .nav-item.dropdown {
    padding: 0;
    text-align: left;
  }
}

.modal-content.rounded-4 {
  border-radius: 2rem;
}

.modal-content .modal-header,
.modal-content .modal-footer {
  border: none;
}

/* Campos e labels alinhados e espaçados */
.modal-body .form-label {
  font-weight: 500;
}

.modal-body .form-control {
  border-radius: 0.5rem;
}

/* Botões maiores e com sombra */
.modal-footer .btn {
  padding: 0.75rem 1.5rem;
  border-radius: 1rem;
  box-shadow: 0 4px 6px rgba(0,0,0,0.1);
}

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\css\login.css

```

/* Cabeçalho azul customizado */
.custom-header {
  background-color: #1659bf;
  color: white;
}

/* Logo */
.logo-img {
  max-height: 100px;
}

/* Botão de login com cor customizada */
.custom-login-btn {
  background-color: #1659bf;
  border-color: #1659bf;
}

/* Botão hover */
.custom-login-btn:hover {
  background-color: #104ba3;
  border-color: #104ba3;
}

/* Card opcional: padding e largura máxima */
.login-card {
  width: 100%;
  max-width: 400px;
}

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\css\style.css

```

/* frontend\static\css\style.css */

/* Reset simples */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: Arial, sans-serif;
  background-color: #f7f9fc;
  display: flex;
  flex-direction: column;
}

header {
  background-color: #0052cc;
  padding: 1rem;
  text-align: center;
  color: white;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

main {
  flex: 1;
  display: flex;
  justify-content: center;
  align-items: center;
}

```



```

}

.login-card {
  background-color: white;
  padding: 2rem;
  border-radius: 2rem;
  box-shadow: 0 8px 16px rgba(0,0,0,0.1);
  width: 320px;
  text-align: center;
  position: relative;
}

.avatar-placeholder {
  width: 100px;
  height: 100px;
  background-color: #e0e0e0;
  border-radius: 1rem;
  margin: 0 auto 1.5rem;
}

.login-card label {
  display: block;
  text-align: left;
  margin-top: 0.5rem;
  font-weight: bold;
}

.login-card input {
  width: 100%;
  padding: 0.5rem;
  margin-top: 0.25rem;
  border: 1px solid #ccc;
  border-radius: 1rem;
}

#recover-link {
  display: inline-block;
  margin: 1rem 0;
  font-size: 0.9rem;
  color: #0052cc;
  text-decoration: none;
}

#login-btn {
  background-color: #0052cc;
  color: white;
  border: none;
  padding: 0.75rem 2rem;
  border-radius: 1rem;
  font-size: 1rem;
  cursor: pointer;
  margin-top: 1rem;
}

#login-btn:hover {
  opacity: 0.9;
}

.list-group-item-action {
  cursor: pointer;
  transition: background-color .2s ease;
}

.list-group-item-action:hover {

```

```
    background-color: #f1f5f9;
  }
}
```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\alertasModule.js

```
// frontend/static/js/alertasModule.js

import { apiService } from './apiService.js';
import { uiService } from './uiService.js';
import { showAlert, formatDateTime, setNewAlertsFlag, updateNotificationBellUI } from './utils.js';

class AlertasModule {
  constructor() {
    this.currentPage = 1;
    this.pageSize = 10;
    this.currentSearchTerm = null;
    this.currentTipoAlerta = null;
    this.currentTotalPages = 1;

    // Armazenar as referências das funções bound uma única vez
    this._boundHandlePaginationClick = this._handlePaginationClick.bind(this);
    this._boundHandlePageSizeChange = this._handlePageSizeChange.bind(this);
    this._boundHandleTableActionClick = this._handleTableActionClick.bind(this);
    this._boundHandleSearchAlert = this._handleSearchAlert.bind(this);
    this._boundHandleClearAlertSearch = this._handleClearAlertSearch.bind(this);
  }

  init() {
    // Nada de especial aqui, os bindings são feitos após a renderização
  }

  async renderAlertsPage() {
    uiService.showLoading();
    try {
      const params = {
        page: this.currentPage,
        size: this.pageSize
      };

      if (this.currentTipoAlerta !== null && this.currentTipoAlerta !== undefined && this.currentTotalPages > 1) {
        params.tipo_alerta = this.currentTipoAlerta;
      }

      if (this.currentSearchTerm !== null && this.currentSearchTerm !== undefined && this.currentTotalPages > 1) {
        params.search_term = this.currentSearchTerm;
      }

      const data = await apiService.get('/alertas/paginated', params);
      this.currentTotalPages = data.total_pages;
      const alerts = data.items;

      const tableHeaders = ['Tipo', 'Mensagem', 'Item ID', 'Data do Alerta', 'Ações'];

      // BLOCO DE FILTROS EM CARD, COM col-12 col-md PARA EMPILHAR ANTES DE ENCOLHER
      const searchAndFilterHtml = `
        <div class="card mb-3">
          <div class="card-header">Filtros de Busca</div>
          <div class="card-body">
            <form id="alertas-search-bar" class="row g-3 mb-0">
              <div class="col-12 col-md">

```

```

        <label for="alert-search-term" class="form-label">Busca</label>
        <input
            type="text"
            id="alert-search-term"
            class="form-control"
            placeholder="Buscar por mensagem ou ID do item"
            value="${this.currentSearchTerm || ''}"
        >
    </div>
    <div class="col-12 col-md">
        <label for="alert-type-filter" class="form-label">Tipo de Alerta</label>
        <select id="alert-type-filter" class="form-select">
            <option value="">Todos os Tipos</option>
            <option value="1" ${this.currentTipoAlerta === 1 ? 'selected' : ''}>Estoque Ba
            <option value="2" ${this.currentTipoAlerta === 2 ? 'selected' : ''}>Validade P
        </select>
    </div>
    <div class="col-12 col-md d-flex justify-content-end align-items-end">
        <button id="btn-search-alert" class="btn btn-primary me-2">Buscar</button>
        <button id="btn-clear-alert-search" class="btn btn-secondary">Limpar</button>
    </div>
</form>
</div>
</div>`;

const tableHtml = uiService.renderTable(tableHeaders, alerts, {
    noRecordsMessage: "Nenhum alerta encontrado.",
    rowMapper: (alerta) => {
        const tipoAlertaText = this._getTipoAlertaText(alerta.tipo_alerta);
        return [
            tipoAlertaText,
            alerta.mensagem_alerta,
            alerta.item_id,
            formatDateTime(alerta.data_alerta)
        ];
    },
    actionsHtml: (alerta) => `
        <div class="d-flex flex-wrap justify-content-center gap-1">
            <button
                class="btn btn-sm btn-info btn-acoes btn-ver-item"
                data-item-id="${alerta.item_id}"
            >
                <i class="bi bi-eye"></i> Ver Item
            </button>
            ${!alerta.ignorar_novos ? `
                <button
                    class="btn btn-sm btn-warning btn-acoes btn-ignorar-alerta"
                    data-id="${alerta.alerta_id}"
                >
                    <i class="bi bi-x-circle"></i> Ignorar Alertas
                </button>` : ``}
            <span class="badge bg-secondary">Ignorado</span>`
            <button
                class="btn btn-sm btn-danger btn-acoes btn-deletar-alerta"
                data-id="${alerta.alerta_id}"
            >
                <i class="bi bi-trash"></i> Deletar
            </button>
        </div>
    `;
});

const paginationHtml = uiService.renderPagination(
    data.page,

```

```

        data.total_pages,
        'alerts', // tipo de paginação para alertas
        'alertsPageSizeSelect', // id do select de pageSize
        this.pageSize
    );

    uiService.renderPage(
        'Lista de Alertas',
        `${searchAndFilterHtml}${tableHtml}${paginationHtml}`
    );

    // Re-anexa os listeners após a renderização do novo conteúdo
    this._bindPageEvents();
    this._bindTableActions();

    // Marcar todos os alertas como visualizados e limpar o sino
    await apiService.markAllAlertsAsViewed();
    setNewAlertsFlag(false);
    updateNotificationBellUI();

    } catch (error) {
        console.error('Erro ao carregar alertas:', error);
        uiService.renderPage(
            'Lista de Alertas',
            `<div class="alert alert-warning">Erro ao carregar alertas: ${error.message} || 'Verifique'`
        );
    } finally {
        uiService.hideLoading();
    }
}

_bindPageEvents() {
    const alertsPaginationNav = document.getElementById('alerts-pagination-nav');
    const pageSizeSelect = document.getElementById('alertsPageSizeSelect');
    const btnSearchAlert = document.getElementById('btn-search-alert');
    const btnClearAlertSearch = document.getElementById('btn-clear-alert-search');

    // Remove previous listeners para evitar duplicação
    if (alertsPaginationNav) {
        alertsPaginationNav.removeEventListener('click', this._boundHandlePaginationClick);
    }

    if (pageSizeSelect) {
        pageSizeSelect.removeEventListener('change', this._boundHandlePageSizeChange);
    }

    if (btnSearchAlert) {
        btnSearchAlert.removeEventListener('click', this._boundHandleSearchAlert);
        btnSearchAlert.addEventListener('click', (e) => {
            e.preventDefault();
            this._boundHandleSearchAlert();
        });
    }

    if (btnClearAlertSearch) {
        btnClearAlertSearch.removeEventListener('click', this._boundHandleClearAlertSearch);
        btnClearAlertSearch.addEventListener('click', (e) => {
            e.preventDefault();
            this._boundHandleClearAlertSearch();
        });
    }

    // Adiciona novos listeners

```

```

        if (alertsPaginationNav) {
            alertsPaginationNav.addEventListener('click', this._boundHandlePaginationClick);
        }
        if (pageSizeSelect) {
            pageSizeSelect.addEventListener('change', this._boundHandlePageSizeChange);
        }
    }

    _handleSearchAlert() {
        this.currentSearchTerm = document.getElementById('alert-search-term').value.trim();
        const selectedType = document.getElementById('alert-type-filter').value;
        this.currentTipoAlerta = selectedType ? parseInt(selectedType) : null;
        this.currentPage = 1;
        this.renderAlertsPage();
    }

    _handleClearAlertSearch() {
        document.getElementById('alert-search-term').value = '';
        document.getElementById('alert-type-filter').value = '';
        this.currentSearchTerm = '';
        this.currentTipoAlerta = null;
        this.currentPage = 1;
        this.renderAlertsPage();
    }

    _handlePaginationClick(e) {
        e.preventDefault();

        // Links de paginação: data-page="alerts-<número>"
        const clickedPageLink = e.target.closest('a[data-page^="alerts-"]');
        if (clickedPageLink) {
            const pageValue = clickedPageLink.dataset.page.split('-')[1];
            const newPage = parseInt(pageValue);
            if (!isNaN(newPage) && newPage !== this.currentPage) {
                this.currentPage = newPage;
                this.renderAlertsPage();
            }
        }
        return;
    }

    // Ações "Anterior / Próximo": data-action="alerts-prev" ou "alerts-next"
    const clickedActionButton = e.target.closest('a[data-action^="alerts-"]');
    if (clickedActionButton) {
        const action = clickedActionButton.dataset.action;
        let newPage = this.currentPage;

        if (action === 'alerts-prev' && newPage > 1) {
            newPage--;
        } else if (action === 'alerts-next' && newPage < this.currentTotalPages) {
            newPage++;
        }

        if (newPage !== this.currentPage) {
            this.currentPage = newPage;
            this.renderAlertsPage();
        }
        return;
    }
}

_handlePageSizeChange(e) {
    this.pageSize = parseInt(e.target.value);

```

```

        this.currentPage = 1;
        this.renderAlertsPage();
    }

    _bindTableActions() {
        const mainContent = document.getElementById('main-content');
        if (mainContent) {
            // Remove e adiciona listener único para clique na tabela
            mainContent.removeEventListener('click', this._boundHandleTableActionClick);
            mainContent.addEventListener('click', this._boundHandleTableActionClick);
        }
    }

    async _handleTableActionClick(e) {
        // Verifica se o botão de ignorar alerta foi clicado
        const ignoreButton = e.target.closest('.btn-ignorar-alerta');
        if (ignoreButton) {
            const alertId = parseInt(ignoreButton.dataset.id);
            if (confirm('Tem certeza que deseja ignorar futuros alertas para este item/motivo?')) {
                await this.ignoreAlert(alertId);
            }
            return; // Retorna após tratar este clique
        }

        // Verifica se o botão de deletar alerta foi clicado
        const deleteButton = e.target.closest('.btn-deletar-alerta');
        if (deleteButton) {
            const alertId = parseInt(deleteButton.dataset.id);
            if (confirm('Tem certeza que deseja DELETAR este alerta? Esta ação é irreversível.')) {
                await this.deleteAlert(alertId);
            }
            return; // Retorna após tratar este clique
        }

        // Verifica se o botão de ver item foi clicado
        const viewItemButton = e.target.closest('.btn-ver-item');
        if (viewItemButton) {
            const itemId = parseInt(viewItemButton.dataset.itemId);
            uiService.showLoading();
            try {
                const itemDetails = await apiService.getItemById(itemId);
                uiService.fillModalDetalhesItem(itemDetails);
                uiService.getModalInstance('modalDetalheItem').show();
            } catch (error) {
                console.error("Erro ao carregar detalhes do item", error);
                showAlert("Não foi possível carregar os detalhes do item.", "danger");
            } finally {
                uiService.hideLoading();
                return; // Retorna após tratar este clique
            }
        }
    }

    async ignoreAlert(alertId) {
        uiService.showLoading();
        try {
            await apiService.patch(`/alertas/ignorar/${alertId}`);
            showAlert(
                'Alerta ignorado com sucesso. Futuros alertas para este item/motivo não serão gerados.',
                'success'
            );
            this.renderAlertsPage();
        } catch (error) {

```

```

        console.error('Erro ao ignorar alerta', error);
        showAlert(error.message || 'Erro ao ignorar o alerta.', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

async deleteAlert(alertId) {
    uiService.showLoading();
    try {
        await apiService.delete(`/alertas/${alertId}`);
        showAlert(
            'Alerta deletado com sucesso.'
        );
        this.renderAlertsPage();
    } catch (error) {
        console.error('Erro ao deletar alerta', error);
        showAlert(error.message || 'Erro ao deletar o alerta.', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

_getTipoAlertaText(tipoAlertaValue) {
    switch (tipoAlertaValue) {
        case 1:
            return '<span class="badge bg-danger"><i class="bi bi-box-fill"></i> Estoque Baixo</span>';
        case 2:
            return '<span class="badge bg-warning text-dark"><i class="bi bi-calendar-x"></i> Validade</span>';
        default:
            return 'Desconhecido';
    }
}

}

export const alertasModule = new AlertasModule();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\apiService.js

```

// frontend/static/js/apiService.js

class ApiService {
    constructor(baseUrl = '/api/almoxarifado') {
        this.baseUrl = baseUrl;
    }

    _getHeaders() {
        const token = localStorage.getItem('token'); // Obter o token mais recente
        const headers = {};
        if (token) {
            headers['Authorization'] = `Bearer ${token}`;
        } else {
            console.warn("Token de autenticação não encontrado no localStorage para a requisição.");
        }
        return headers;
    }

    async _fetch(endpoint, options = {}) {
        const url = `${this.baseUrl}${endpoint}`;
        const requestHeaders = {

```

```

        ...this._getHeaders(),
        ...options.headers // Mescla com quaisquer headers específicos da requisição
    };
    options.headers = requestHeaders; // Atribui os headers mesclados de volta às opções

    if (options.body instanceof FormData) {
        delete options.headers['Content-Type'];
    } else if (!options.headers['Content-Type']) {
        options.headers['Content-Type'] = 'application/json';
    }

    try {
        const response = await fetch(url, {
            ...options,
            headers: options.headers // Use the prepared headers
        });

        if (!response.ok) {
            // Se a resposta for 204 No Content, não tente parsear JSON
            if (response.status === 204) {
                return {}; // Retorna um objeto vazio para 204
            }
            const errorData = await response.json().catch(() => ({ detail: 'Erro desconhecido na res
            throw new Error(errorData.detail || `Erro na API: ${response.status} ${response.statusTe

        }

        // Se a resposta for 204 No Content, não tente parsear JSON
        if (response.status === 204) {
            return {};
        }

        return response.json();

    } catch (error) {
        console.error(`Falha na requisição para ${url}:`, error);
        throw error;
    }
}

async get(endpoint, params = {}) {
    const queryString = new URLSearchParams(params).toString();
    const urlWithParams = `${endpoint}${queryString ? `?${queryString}` : ''}`;
    return await this._fetch(urlWithParams, { method: 'GET' });
}

async post(endpoint, data) {
    return this._fetch(endpoint, {
        method: 'POST',
        body: JSON.stringify(data)
    });
}

async put(endpoint, data) {
    return this._fetch(endpoint, {
        method: 'PUT',
        body: JSON.stringify(data)
    });
}

async patch(endpoint, data = {}) {
    return this._fetch(endpoint, {
        method: 'PATCH',
        body: JSON.stringify(data)
    });
}

```



```

    });
  }

  async delete(endpoint) {
    return this._fetch(endpoint, {
      method: 'DELETE'
    });
  }

  // Métodos específicos

  async getUsuarioById(id) {
    try {
      const user = await this.get(`/usuarios/${id}`);
      return user.nome_usuario;
    } catch (e) {
      return `#${id}`;
    }
  }

  async getSetorById(id) {
    try {
      const setor = await this.get(`/setores/${id}`);
      return setor.nome_setor;
    } catch (e) {
      return `#${id}`;
    }
  }

  async fetchAllRetiradas(page, pageSize, filters = {}) {
    const params = { page, page_size: pageSize };
    const queryParamsForApi = {};

    if (filters.status !== null && filters.status !== undefined && filters.status !== '') {
      queryParamsForApi.status = filters.status;
    }
    if (filters.solicitante) {
      queryParamsForApi.solicitante = filters.solicitante;
    }
    if (filters.start_date) {
      queryParamsForApi.start_date = filters.start_date;
    }
    if (filters.end_date) {
      queryParamsForApi.end_date = filters.end_date;
    }

    const hasActiveFilters = Object.keys(queryParamsForApi).length > 0;
    const endpoint = hasActiveFilters ? '/retiradas/search' : '/retiradas/paginated';
    const responseData = await this.get(endpoint, { ...params, ...queryParamsForApi });

    return {
      current_page: responseData.page,
      total_pages: responseData.pages,
      total_items: responseData.total,
      items: responseData.items
    };
  }

  async fetchRetiradasPendentes(page, pageSize) {
    const responseData = await this.get('/retiradas/pendentes/paginated', { page, page_size: pageSize });
    return {
      current_page: responseData.page,
      total_pages: responseData.pages,
    };
  }

```

```

        total_items: responseData.total,
        items: responseData.items
    };
}

async updateRetiradaStatus(id, status, detail) {
    return this.put(`/retiradas/${id}`, { status, detalhe_status: detail });
}

async fetchAllItens() {
    return this.get('/itens');
}

async fetchAllSetores() {
    return this.get('/setores');
}

async solicitarRetirada(data) {
    return this.post('/retiradas/', data);
}

async searchItems(nome = null, categoria = null, page = 1, size = 10) {
    const params = { page, size };
    if (nome) {
        params.nome = nome;
    }
    if (categoria) {
        params.categoria = categoria;
    }
    return this.get('/itens/buscar', params);
}

async getItemById(itemId) {
    return this.get(`/itens/${itemId}`);
}

async getUnviewedAlertsCount() {
    try {
        const response = await this.get('/alertas/unviewed-count');
        return response.count;
    } catch (error) {
        console.error('Erro ao buscar contagem de alertas não visualizados', error);
        return 0;
    }
}

async markAllAlertsAsViewed() {
    try {
        await this.patch('/alertas/mark-viewed');
    } catch (error) {
        console.error('Erro ao marcar alertas como visualizados', error);
        throw error;
    }
}

async uploadBulkItems(file) {
    const formData = new FormData();
    formData.append('file', file);
    return this._fetch('/itens/upload-bulk/', {
        method: 'POST',
        body: formData,
    });
}

```

```

    async fetchUserRetiradasPaginated(page, pageSize) {
      const responseData = await this.get('/retiradas/minhas-retiradas/paginated', { page, page_size:
      return {
        current_page: responseData.page,
        total_pages: responseData.pages,
        total_items: responseData.total,
        items: responseData.items
      };
    }

    // Retorna o ID do setor do usuário também
    async getCurrentUserDetails(userId) {
      try {
        const user = await this.get(`/usuarios/${userId}`);
        const sectorName = await this.getSetorById(user.setor_id);
        return {
          name: user.nome_usuario,
          siape: user.siape_usuario,
          sectorName: sectorName,
          sectorId: user.setor_id // ADICIONADO: ID do setor do usuário
        };
      } catch (error) {
        console.error("Erro ao carregar detalhes do usuário ou setor:", error);
        return { name: 'Usuário', siape: 'N/A', sectorName: 'N/A', sectorId: null }; // Adicionado s
      }
    }
  }
}

export const apiService = new ApiService();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxa rifado_ets\frontend\static\js\bootstrap.bundle.min.js

```

/*!
 * Bootstrap v5.3.5 (https://getbootstrap.com/)
 * Copyright 2011-2025 The Bootstrap Authors (https://github.com/twbs/bootstrap/graphs/contributors)
 * Licensed under MIT (https://github.com/twbs/bootstrap/blob/main/LICENSE)
 */
!function(t,e){"object"==typeof exports&&"undefined"!=typeof module?module.exports=e():"function"==typeof
}
//# sourceMappingURL=bootstrap.bundle.min.js.map

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxa rifado_ets\frontend\static\js\cadastrar-categorias.js

```

// frontend/static/js/cadastrar-categorias.js

document.addEventListener('DOMContentLoaded', () => {
  const modalEl = document.getElementById('modalCadastrarCategoria');
  const modal = new bootstrap.Modal(modalEl);
  const form = document.getElementById('form-cadastrar-categoria');

  // Abrir
  document.getElementById('btn-open-cadastrar-categoria')?.addEventListener('click', e => {
    e.preventDefault(); form.reset(); modal.show();
  });
  document.getElementById('btn-cancelar-categoria')?.addEventListener('click', () => {
    modal.hide(); document.querySelectorAll('.modal-backdrop').forEach(el=>el.remove());
  });
});

```

```

// Salvar cadastro
document.getElementById('btn-salvar-categoria').onclick = async () => {
  if(!form.checkValidity()) return form.reportValidity();
  const data = { nome_categoria: form.nome_categoria.value, descricao_categoria: form.descricao_categoria.value };
  const token = localStorage.getItem('token');
  const res = await fetch('/api/almoxarifado/categorias/', {
    method: 'POST', headers: { 'Content-Type': 'application/json', 'Authorization': `Bearer ${token}` }, body: JSON.stringify(data)
  });
  if(!res.ok) return alert('Erro ao cadastrar categoria');

  form.reset(); modal.hide(); document.querySelectorAll('.modal-backdrop').forEach(el=>el.remove());
  renderizarCategorias();

  // alerta de sucesso
  const alertBox = document.getElementById('alert-categoria');
  alertBox.innerHTML = `<div class="alert alert-success alert-dismissible fade show" role="alert">Categoria cadastrada com sucesso!</div>`;
  setTimeout(()=>{ const a = alertBox.querySelector('.alert'); if(a) bootstrap.Alert.getOrCreateInstance(a).close(); }, 3000);
};

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\cadastrar-item.js

```

//frontend\static\js\cadastrar-item.js

document.addEventListener('DOMContentLoaded', () => {
  const modalEl = document.getElementById('modalCadastrarItem');
  const modal = new bootstrap.Modal(modalEl);

  async function preencherCategoriasNoCadastro() {
    const token = localStorage.getItem('token');
    const resp = await fetch('/api/almoxarifado/categorias', {
      headers: { 'Authorization': `Bearer ${token}`, 'Accept': 'application/json' }
    });
    if (!resp.ok) {
      console.error('[Cadastro] Erro ao carregar categorias:', resp.status);
      return;
    }
    const cats = await resp.json();

    // Busca o <select> dentro do próprio modal
    const sel = modalEl.querySelector('select[name="categoria_id"]');
    if (!sel) return console.error('[Cadastro] select[name="categoria_id"] não encontrado!');

    sel.innerHTML = '<option value="" disabled selected>Selecione...</option>';
    cats.forEach(c => {
      const o = document.createElement('option');
      o.value = c.categoria_id;
      o.textContent = `${c.categoria_id} - ${c.nome_original.toUpperCase()}`;
      sel.append(o);
    });
  }

  // Use shown.bs.modal para garantir que o modal já está completamente aberto
  modalEl.addEventListener('shown.bs.modal', preencherCategoriasNoCadastro);

  // resto do código continua igual...
  document.querySelectorAll('#btn-open-cadastrar-item').forEach(btn => {
    btn.addEventListener('click', e => {
      e.preventDefault();
      modal.show();
    });
  });
});

```

```

    });
  });

async function preencherCategoriasNoCadastro() {
  const token = localStorage.getItem('token');
  const resp = await fetch('/api/almoxarifado/categorias', {
    headers: { 'Authorization': `Bearer ${token}`, 'Accept': 'application/json' }
  });
  if (!resp.ok) return console.error('Falha ao carregar categorias');
  const cats = await resp.json();
  const sel = document.getElementById('categoria_id');
  sel.innerHTML = '<option value="" disabled selected>Selecione...</option>';
  cats.forEach(c => {
    const o = document.createElement('option');
    o.value = c.categoria_id;
    o.textContent = `${c.categoria_id} - ${c.nome_categoria.toUpperCase()}`;
    sel.append(o);
  });
}

modalEl.addEventListener('show.bs.modal', preencherCategoriasNoCadastro);

const form = document.getElementById('form-cadastrar-item');
const btnSalvar = document.getElementById('btn-salvar-item');

// 1) Cancelar
document.getElementById('btn-cancelar-item').addEventListener('click', () => {
  modal.hide();
});

// 2) Ao esconder o modal, limpa backdrop e classe no body
modalEl.addEventListener('hidden.bs.modal', () => {
  // remove sombra e bloqueio de clique
  document.body.classList.remove('modal-open');
  document.querySelectorAll('.modal-backdrop').forEach(el => el.remove());
});

btnSalvar.addEventListener('click', async () => {
  if (!form.checkValidity()) {
    form.reportValidity();
    return;
  }

  // monta objeto e filtra valores vazios
  const formData = new FormData(form);
  const data = {};
  for (const [key, value] of formData.entries()) {
    if (value !== '') {
      // converte numéricos
      if (['quantidade_item', 'quantidade_minima_item', 'categoria_id'].includes(key)) {
        data[key] = Number(value);
      } else {
        data[key] = value;
      }
    }
  }
  // formata data_entrada_item se existir
  if (data.data_entrada_item) {
    data.data_entrada_item = new Date(data.data_entrada_item).toISOString();
  }
  // data_validade_item já vem em yyyy-mm-dd e o Pydantic aceita

  try {

```

```

const token = localStorage.getItem('token');
const resp = await fetch('/api/almoxarifado/itens/', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`
  },
  body: JSON.stringify(data)
});

if (resp.status === 201) {
  alert('✓ Item cadastrado com sucesso!');
  form.reset();
  modal.hide();
  // atualiza listagem se aberta
  if (typeof renderizarListaItens === 'function') renderizarListaItens();
} else {
  // exibe erros vindo da API
  const err = await resp.json();
  if (Array.isArray(err.detail)) {
    // Pydantic retorna lista de erros
    const msgs = err.detail.map(e => `${e.loc.join('.')}: ${e.msg}`);
    alert('Erros:\n' + msgs.join('\n'));
  } else {
    alert(err.detail || 'Falha ao cadastrar item.');
  }
}
} catch (e) {
  console.error(e);
  alert('Erro de conexão com o servidor.');
```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\dataService.js

```

// frontend/static/js/dataService.js
import { apiService } from '../apiService.js';
import estadoGlobal from '../estadoGlobal.js';

class DataService {
  async buildLookupMaps(retiradas) {
    const userIds = [...new Set(retiradas.map(r => r.usuario_id).filter(id => id != null))];
    // Coleta IDs de usuários que autorizaram também
    const adminIds = [...new Set(retiradas.map(r => r.autorizado_por).filter(id => id != null))];
    const allUserIds = [...new Set([...userIds, ...adminIds])]; // Combina e remove duplicatas

    const setorIds = [...new Set(retiradas.map(r => r.setor_id).filter(id => id != null))];

    const [usuariosArr, setoresArr] = await Promise.all([
      Promise.all(allUserIds.map(id => apiService.getUsuarioById(id).then(name => [id, name]))),
      Promise.all(setorIds.map(id => apiService.getSetorById(id).then(name => [id, name]))
    ]);

    return {
      usuarioMap: Object.fromEntries(usuariosArr), // Este mapa agora contém todos os nomes de usu
      setorMap: Object.fromEntries(setoresArr)
    };
  }

  async getProcessedRetiradas(fetchFunction, page, pageSize, filters = {}) {
```

```

const data = await fetchFunction(page, pageSize, filters);
const { usuarioMap, setorMap } = await this.buildLookupMaps(data.items);

const processedItems = data.items.map(r => ({
  ...r,
  usuario_nome: r.solicitado_localmente_por || usuarioMap[r.usuario_id] || 'N/A',
  setor_nome: setorMap[r.setor_id] || 'N/A',
  autorizado_por_nome: r.autorizado_por ? usuarioMap[r.autorizado_por] : 'N/A' // Adiciona o nome
})));

return {
  ...data,
  items: processedItems
};
}

export const dataService = new DataService();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\estadoGlobal.js

```

// frontend\static\js\estadoGlobal.js

const estadoGlobal = {
  allRetiradas: [],
  pendentesRetiradas: [],
  currentHistoricoPage: 1,
  totalHistoricoPages: 1,
  currentHistoricoPageSize: 10,
  currentHistoricoFilters: {},

  currentPendentesPage: 1,
  totalPendentesPages: 1,
  currentPendentesPageSize: 10,

  // NOVO: Estado para o histórico de retiradas do servidor
  currentMinhasRetiradasPage: 1,
  totalMinhasRetiradasPages: 1,
  currentMinhasRetiradasPageSize: 10,

  PAGE_SIZE_OPTIONS: [5, 10, 25, 50, 100],

  statusMap: { // converter de INT para STRING (para exibição)
    1: 'PENDENTE',
    2: 'AUTORIZADA',
    3: 'CONCLUÍDA',
    4: 'NEGADA'
  },
  statusMapUpdate: { // converter de STRING para INT (para envio à API)
    PENDENTE: 1,
    AUTORIZADA: 2,
    CONCLUIDA: 3,
    NEGADA: 4
  },

  // Métodos para atualizar o estado de forma controlada

  setHistoricoPagination(currentPage, totalPages, pageSize, filters) {
    this.currentHistoricoPage = currentPage;
    this.totalHistoricoPages = totalPages;
  }
};

```

```

        this.currentHistoricoPageSize = pageSize;
        this.currentHistoricoFilters = { ...filters };
    },

    setPendentesPagination(currentPage, totalPages, pageSize) {
        this.currentPendentesPage = currentPage;
        this.totalPendentesPages = totalPages;
        this.currentPendentesPageSize = pageSize;
    },

    // Método para atualizar o estado de paginação das minhas retiradas
    setMinhasRetiradasPagination(currentPage, totalPages, pageSize) {
        this.currentMinhasRetiradasPage = currentPage;
        this.totalMinhasRetiradasPages = totalPages;
        this.currentMinhasRetiradasPageSize = pageSize;
    },

    setAllRetiradas(data) {
        this.allRetiradas = data;
    },

    setPendentesRetiradas(data) {
        this.pendentesRetiradas = data;
    },

    // Método para definir as minhas retiradas
    setMinhasRetiradas(data) {
        // Pode ser um array separado ou parte de allRetiradas se o filtro for sempre aplicado
        // Para simplificar e manter a separação, vou usar um novo array.
        this.minhasRetiradas = data;
    }
};

export default estadoGlobal;

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\historicoServidorModule.js

```

// frontend/static/js/historicoServidorModule.js

import { apiService } from './apiService.js';
import { dataService } from './dataService.js';
import { uiService } from './uiService.js';
import { showAlert, getStatusText, formatDateTime } from './utils.js';
import estadoGlobal from './estadoGlobal.js';

class HistoricoServidorModule {
    constructor() {
        this.minhasRetiradasPageSizeSelectId = 'minhasRetiradasPageSize';

        // Bindings para eventos que são removidos e adicionados novamente
        this.boundHandleMinhasRetiradasPaginationClick = this._handleMinhasRetiradasPaginationClick.bind(this);
        this.boundHandleMinhasRetiradasPageSizeChange = this._handleMinhasRetiradasPageSizeChange.bind(this);
        this.boundHandleDetalhesRetiradaClick = this._handleDetalhesRetiradaClick.bind(this);
    }

    init() {
        // Nenhuma inicialização especial aqui, os eventos serão vinculados após a renderização
    }

    async renderMinhasRetiradas(page = 1, pageSize = estadoGlobal.currentMinhasRetiradasPageSize) {

```



```

uiService.showLoading();
try {
  // Usa a nova função da apiService para buscar as retiradas do usuário logado
  const data = await dataService.getProcessedRetiradas(apiService.fetchUserRetiradasPaginated);

  estadoGlobal.setMinhasRetiradasPagination(data.current_page, data.total_pages, pageSize);
  estadoGlobal.setMinhasRetiradas(data.items);

  const tableHeaders = ['ID', 'Data', 'Status', 'Itens', 'Ações'];

  const tableHtml = uiService.renderTable(tableHeaders, estadoGlobal.minhasRetiradas, {
    noRecordsMessage: "Nenhuma retirada encontrada.",
    rowMapper: (r) => [
      r.retirada_id,
      formatDateTime(r.data_solicitacao),
      getStatusText(r.status),
      // Exibe uma lista de itens solicitados
      r.itens.map(item => `${item.item.nome_item_original} (${item.quantidade_retirada})`),
    ],
    actionsHtml: (r) => `
      <div class="d-flex flex-wrap justify-content-center gap-1">
        <button
          class="btn btn-sm btn-info btn-acoes btn-detalhes-minha-retirada"
          data-id="${r.retirada_id}"
        >
          <i class="bi bi-eye"></i> Detalhes
        </button>
      </div>
    `;
  });

  const paginationHtml = uiService.renderPagination(
    estadoGlobal.currentMinhasRetiradasPage,
    estadoGlobal.totalMinhasRetiradasPages,
    'minhas-retiradas', // Tipo de paginação para minhas retiradas
    this.minhasRetiradasPageSizeSelectId,
    estadoGlobal.currentMinhasRetiradasPageSize
  );

  uiService.renderPage('Meu Histórico de Retiradas', `
    ${tableHtml}
    ${paginationHtml}
  `);

  this._bindMinhasRetiradasEvents();
  this._bindMinhasRetiradasTableActions();

} catch (error) {
  console.error("Erro ao renderizar histórico de retiradas do servidor:", error);
  showAlert(error.message || 'Ocorreu um erro ao carregar seu histórico de retiradas.', 'danger');
} finally {
  uiService.hideLoading();
}

_bindMinhasRetiradasEvents() {
  const minhasRetiradasPaginationNav = document.getElementById('minhas-retiradas-pagination-nav');
  const pageSizeSelect = document.getElementById(this.minhasRetiradasPageSizeSelectId);

  if (minhasRetiradasPaginationNav) {
    minhasRetiradasPaginationNav.removeEventListener('click', this.boundHandleMinhasRetiradasPagina
    minhasRetiradasPaginationNav.addEventListener('click', this.boundHandleMinhasRetiradasPagina
  }
}

```

```

    if (pageSizeSelect) {
      pageSizeSelect.removeEventListener('change', this.boundHandleMinhasRetiradasPageSizeChange);
      pageSizeSelect.addEventListener('change', this.boundHandleMinhasRetiradasPageSizeChange);
    }
  }

  _handleMinhasRetiradasPaginationClick(e) {
    e.preventDefault();

    const clickedPageLink = e.target.closest('a[data-page^="minhas-retiradas-"]');
    const clickedActionButton = e.target.closest('a[data-action^="minhas-retiradas-"]');

    if (clickedPageLink) {
      const pageValue = clickedPageLink.dataset.page.split('-')[2]; // Ajustado para pegar o número
      const newPage = parseInt(pageValue);

      if (!isNaN(newPage) && newPage !== estadoGlobal.currentMinhasRetiradasPage) {
        this.renderMinhasRetiradas(newPage, estadoGlobal.currentMinhasRetiradasPageSize);
      }
      return;
    }

    if (clickedActionButton) {
      const action = clickedActionButton.dataset.action;
      let newPage = estadoGlobal.currentMinhasRetiradasPage;

      if (action === 'minhas-retiradas-prev' && newPage > 1) {
        newPage--;
      } else if (action === 'minhas-retiradas-next' && newPage < estadoGlobal.totalMinhasRetiradas) {
        newPage++;
      }

      if (newPage !== estadoGlobal.currentMinhasRetiradasPage) {
        this.renderMinhasRetiradas(newPage, estadoGlobal.currentMinhasRetiradasPageSize);
      }
      return;
    }
  }

  _handleMinhasRetiradasPageSizeChange(e) {
    if (e.target.id === this.minhasRetiradasPageSizeSelectId) {
      const newPageSize = parseInt(e.target.value);
      this.renderMinhasRetiradas(1, newPageSize);
    }
  }

  _bindMinhasRetiradasTableActions() {
    const mainContent = document.getElementById('main-content');
    if (mainContent) {
      // Remove e adiciona listener único para clique na tabela
      mainContent.removeEventListener('click', this.boundHandleDetalhesRetiradaClick);
      mainContent.addEventListener('click', this.boundHandleDetalhesRetiradaClick);
    }
  }

  _handleDetalhesRetiradaClick(e) {
    const detailsButton = e.target.closest('.btn-detalhes-minha-retirada');
    if (detailsButton) {
      const id = parseInt(detailsButton.dataset.id);
      const retirada = estadoGlobal.minhasRetiradas.find(r => r.retirada_id === id);
      if (retirada) {
        uiService.fillModalDetalhes(retirada);
        uiService.getModalInstance('modalVerDetalhesRetirada').show();
      }
    }
  }

```

```
export const historicoServidorModule = new HistoricoServidorModule();
```

```
// frontend/static/js/listar-categorias.js
```

```

        <label for="search-categoria-nome" class="form-label">Nome da Categoria</label>
        <input
            type="text"
            id="search-categoria-nome"
            class="form-control"
            placeholder="Buscar por nome"
            value="{searchCat}"
        >
    </div>

    <!-- Coluna 2: botões empilham em telas < md -->
    <div class="col-12 col-md d-flex justify-content-end align-items-end">
        <button type="button" id="btn-search-cat" class="btn btn-primary me-2">Buscar</button>
        <button type="button" id="btn-clear-search-cat" class="btn btn-secondary">Limpar</button>
    </div>

    </form>
</div>
</div>

<!-- Tabela de Categorias -->
<div class="table-responsive">
    <table class="table table-bordered table-striped">
        <thead class="table-secondary text-center">
            <tr>
                <th>ID</th>
                <th>Nome</th>
                <th>Descrição</th>
                <th>Ações</th>
            </tr>
        </thead>
        <tbody>
            {data.items.map((c) => `
                <tr>
                    <td class="text-center">{c.categoria_id}</td>
                    <td>{c.nome_original}</td>
                    <td>{c.descricao_categoria || '-'}</td>
                    <td class="text-center">
                        <!-- Container flex para igualar largura e espaçar verticalmente -->
                        <div class="d-flex flex-wrap justify-content-center gap-1">
                            <button
                                class="btn btn-sm btn-primary btn-acoes btn-editar-cat"
                                data-id="{c.categoria_id}"
                            >
                                Editar
                            </button>
                            <button
                                class="btn btn-sm btn-danger btn-acoes btn-deletar-cat"
                                data-id="{c.categoria_id}"
                            >
                                Deletar
                            </button>
                        </div>
                    </td>
                </tr>
            `).join('')}</tbody>
        </table>
    </div>

    <!-- Paginação -->
    <nav>
        <ul class="pagination justify-content-center">
            <li class="page-item {currentPageCat === 1 ? 'disabled' : ''}">

```

```

        <a class="page-link" href="#" data-action="prev-cat">Anterior</a>
    </li>
    ${(() => {
        const pages = [];
        const start = Math.max(1, currentPageCat - 2);
        const end = Math.min(data.total_pages, currentPageCat + 2);
        for (let p = start; p <= end; p++) {
            pages.push(`
                <li class="page-item ${p === currentPageCat ? 'active' : ''}">
                    <a class="page-link" href="#" data-page-cat="${p}">${p}</a>
                </li>`);
        }
        return pages.join('');
    })()}
    <li class="page-item ${currentPageCat === data.total_pages ? 'disabled' : ''}">
        <a class="page-link" href="#" data-action="next-cat">Próximo</a>
    </li>
</ul>
</nav>

<!-- Controle de itens por página -->
<div class="d-flex justify-content-center my-2">
    <label class="me-2">Categorias por página:</label>
    <select id="page-size-cat" class="form-select w-auto">
        ${[5,10,25,50,100].map(opt => `
            <option value="${opt}" ${opt === pageSizeCat ? 'selected' : ''}>${opt}</option>`
        )}.join('')}
    </select>
</div>
`;

bindCategoriaActions();
bindPaginationCat(data.total_pages);
}

// Ações
function bindCategoriaActions() {
    const main = document.getElementById('main-content');

    document.getElementById('btn-search-cat').onclick = (e) => {
        e.preventDefault();
        searchCat = document.getElementById('search-categoria-nome').value.trim();
        currentPageCat = 1;
        renderizarCategorias();
    };

    document.getElementById('btn-clear-search-cat').onclick = (e) => {
        e.preventDefault();
        searchCat = '';
        currentPageCat = 1;
        renderizarCategorias();
    };

    main.querySelectorAll('.btn-editar-cat').forEach(btn => {
        btn.onclick = async () => {
            const id = btn.dataset.id;
            const token = localStorage.getItem('token');
            const res = await fetch(`/api/almoxarifado/categorias/${id}`, {
                headers: {'Authorization': `Bearer ${token}`, 'Accept': 'application/json'}
            });
            if (!res.ok) return alert('Erro ao carregar categoria');
            const cat = await res.json();
            // Popula modal de edição

```

```

        document.getElementById('edit-nome_categoria').value = cat.nome_original;
        document.getElementById('edit-descricao_categoria').value = cat.descricao_categoria || '';
        const saveBtn = document.getElementById('btn-salvar-editar-categoria');
        saveBtn.dataset.id = id;

        // Exibe modal
        new bootstrap.Modal(document.getElementById('modalEditarCategoria')).show();
    };
});

main.querySelectorAll('.btn-deletar-cat').forEach(btn => {
    btn.onclick = async () => {
        if (!confirm('Excluir categoria?')) return;

        const id = btn.dataset.id;
        const token = localStorage.getItem('token');
        const resp = await fetch(
            `/api/almoxarifado/categorias/${id}`,
            { method: 'DELETE', headers: { 'Authorization': `Bearer ${token}` } }
        );

        if (!resp.ok) {
            // Tenta ler o detalhe retornado pelo FastAPI
            let err;
            try {
                const body = await resp.json();
                err = body.detail || resp.statusText;
            } catch {
                err = resp.statusText;
            }
            return alert(`Erro ao excluir categoria:\n${err}`);
        }

        // Só re-renderiza se deu sucesso
        renderizarCategorias();
    };
});
}

// Salvar edição
document.getElementById('btn-salvar-editar-categoria').onclick = async e => {
    const id = e.currentTarget.dataset.id;
    if (!id) return alert('ID não definido');
    const form = document.getElementById('form-editar-categoria');
    if (!form.checkValidity()) return form.reportValidity();

    const data = {
        nome_categoria: form.nome_categoria.value,
        descricao_categoria: form.descricao_categoria.value || undefined
    };
    const token = localStorage.getItem('token');
    const res = await fetch(`/api/almoxarifado/categorias/${id}`, {
        method: 'PUT',
        headers: { 'Content-Type': 'application/json', 'Authorization': `Bearer ${token}` },
        body: JSON.stringify(data)
    });
    if (!res.ok) return alert('Erro ao salvar categoria');

    bootstrap.Modal.getInstance(document.getElementById('modalEditarCategoria')).hide();
    renderizarCategorias();
};

// Paginação

```

```

function bindPaginationCat(totalPages) {
  const main = document.getElementById('main-content');
  main.querySelector('[data-action="prev-cat"]').onclick = e => {
    e.preventDefault();
    if (currentPageCat > 1) {
      currentPageCat--;
      renderizarCategorias();
    }
  };
  main.querySelector('[data-action="next-cat"]').onclick = e => {
    e.preventDefault();
    if (currentPageCat < totalPages) {
      currentPageCat++;
      renderizarCategorias();
    }
  };
  main.querySelectorAll('[data-page-cat]').forEach(el => {
    el.onclick = e => {
      e.preventDefault();
      currentPageCat = +el.dataset.pageCat;
      renderizarCategorias();
    };
  });
  document.getElementById('page-size-cat').onchange = e => {
    pageSizeCat = +e.target.value;
    currentPageCat = 1;
    renderizarCategorias();
  };
}

// Inicialização
document.getElementById('listar-categoria-link')?.addEventListener('click', e => {
  e.preventDefault();
  currentPageCat = 1;
  renderizarCategorias();
});
document.getElementById('listar-categoria-link-quick')?.addEventListener('click', e => {
  e.preventDefault();
  currentPageCat = 1;
  renderizarCategorias();
});
}

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\listar-itens.js

```

// frontend/static/js/listar-itens.js

// Estado de paginação
let currentPage = 1;
let pageSize = 10;
const pageSizeOptions = [5, 10, 25, 50, 100];

// guarda estado da última busca
let searchNome = '';
let searchCategoria = '';

// 1) API de categorias
async function carregarCategorias() {
  const token = localStorage.getItem('token');
  const resp = await fetch("/api/almoxarifado/categorias", {
    headers: { 'Authorization': `Bearer ${token}`, 'Accept': 'application/json' }
  });
}

```

```

    });
    if (resp.status === 401) {
        window.location = '/';
        return [];
    }
    if (!resp.ok) throw new Error(`HTTP ${resp.status}`);
    return resp.json();
}

//2) API de itens
async function carregarListItens(page = currentPage, size = pageSize) {
    const token = localStorage.getItem('token');
    const resp = await fetch(`/api/almoxarifado/itens/paginated?page=${page}&size=${size}`, {
        headers: { 'Authorization': `Bearer ${token}`, 'Accept': 'application/json' }
    });
    if (resp.status === 401) {
        window.location = '/';
        return { items: [], total_pages: 0 };
    }
    if (!resp.ok) throw new Error(`HTTP ${resp.status}`);
    const data = await resp.json();
    // Ordena itens alfabeticamente por nome
    data.items.sort((a, b) => a.nome_item_original.localeCompare(b.nome_item_original));
    return data;
}

async function buscarItens(nome, categoria, page = currentPage, size = pageSize) {
    const token = localStorage.getItem('token');
    const params = new URLSearchParams({ page, size });
    if (nome) params.append('nome', nome);
    if (categoria) params.append('categoria', categoria);

    const resp = await fetch(`/api/almoxarifado/itens/buscar?${params}`, {
        headers: { 'Authorization': `Bearer ${token}`, 'Accept': 'application/json' }
    });
    if (resp.status === 401) {
        window.location = '/';
        return { items: [], total_pages: 0 };
    }
    if (!resp.ok) throw new Error(`HTTP ${resp.status}`);
    const data = await resp.json();
    // Ordena itens da busca
    data.items.sort((a, b) => a.nome_item_original.localeCompare(b.nome_item_original));
    return data;
}

//3) Templates HTML
function criarSearchBar() {
    return `
        <h3 class="mb-3">Lista de Itens</h3>
        <div class="card mb-3">
            <div class="card-header">Filtros de Busca</div>
            <div class="card-body">
                <form id="search-bar" class="row g-3 mb-0">
                    <div class="col-12 col-md">
                        <label for="search-nome" class="form-label">Nome</label>
                        <input type="text" id="search-nome" class="form-control" placeholder="Buscar por">
                    </div>
                    <div class="col-12 col-md">
                        <label for="search-categoria" class="form-label">Categoria</label>
                        <input type="text" id="search-categoria" class="form-control" placeholder="Busca">
                    </div>
                    <div class="col-12 col-md d-flex justify-content-end align-items-end">

```



```

        <button id="btn-search" class="btn btn-primary me-2">Buscar</button>
        <button id="btn-clear-search" class="btn btn-secondary">Limpar</button>
    </div>
</form>
</div>
</div>
};
}

// MODIFICADO: Adicionado parâmetro isReadOnly
function criarTabelaltens(itens, categoryMap, isReadOnly) {
    let html = `
        <div class="table-responsive">
            <table class="table table-bordered table-striped">
                <thead class="table-secondary text-center">
                    <tr>
                        <th>ID</th>
                        <th>Nome</th>
                        <th>Descrição</th>
                        <th>Unidade de Medida</th>
                        <th>Quantidade</th>
                        <th>Validade</th>
                        <th>Entrada</th>
                        <th>Marca</th>
                        <th>Categoria</th>
                        ${!isReadOnly ? '<th>Ações</th>' : ''}
                    </tr>
                </thead>
                <tbody>

itens.forEach(item => {
    const cat = categoryMap[item.categoria_id];
    const label = cat
        ? `${item.categoria_id} ${cat.nome_original.toUpperCase()}`
        : item.categoria_id;

    const dataValidade = item.data_validade_item
        ? new Date(item.data_validade_item).toLocaleDateString()
        : '-';

    const dataEntrada = item.data_entrada_item
        ? new Date(item.data_entrada_item).toLocaleDateString()
        : '-';

    html += `
        <tr>
            <td>${item.item_id}</td>
            <td>${item.nome_item_original}</td>
            <td>${item.descricao_item || '-'}</td>
            <td>${item.unidade_medida_item}</td>
            <td class="text-center">${item.quantidade_item}</td>
            <td class="text-center">${dataValidade}</td>
            <td class="text-center">${dataEntrada}</td>
            <td>${item.marca_item || '-'}</td>
            <td class="text-center">${label}</td>
            ${!isReadOnly ? `
                <td class="text-center">
                    <div class="d-flex flex-wrap justify-content-center gap-1">
                        <button class="btn btn-sm btn-primary btn-acoes btn-editar" data-id="${item.`
                    <i class="bi bi-pencil-square"></i> Editar
                    </button>
                        <button class="btn btn-sm btn-danger btn-acoes btn-deletar" data-id="${item.`
    `;
}

```

```

                <i class="bi bi-trash"></i> Deletar
            </button>
        </div>
    </td>
    ` : ''}
</tr>
`;
});

html += `
        </tbody>
    </table>
</div>
`;
return html;
}

function criarControlesPaginacao(totalPages) {
    let html = `
        <nav aria-label="Page navigation" id="itens-pagination-nav">
            <ul class="pagination justify-content-center">
                <li class="page-item ${currentPage === 1 ? 'disabled' : ''}>
                    <a class="page-link" href="#" data-action="itens-prev">Anterior</a>
                </li>
            </ul>
        `;

    const start = Math.max(1, currentPage - 2);
    const end = Math.min(totalPages, currentPage + 2);

    for (let p = start; p <= end; p++) {
        html += `
            <li class="page-item ${p === currentPage ? 'active' : ''}>
                <a class="page-link" href="#" data-page="itens-${p}">${p}</a>
            </li>
        `;
    }

    html += `
        <li class="page-item ${currentPage === totalPages || totalPages === 0 ? 'disabled' : ''}>
            <a class="page-link" href="#" data-action="itens-next">Próximo</a>
        </li>
    </ul>
    </nav>
    <div class="d-flex justify-content-center my-2">
        <label class="me-2">Itens por página: </label>
        <select id="page-size-select" class="form-select w-auto">
            ${pageSizeOptions
                .map(opt => `<option value="${opt}" ${opt === pageSize ? 'selected' : ''}>${opt}</option>`
                .join('')}
        </select>
    </div>
    `;
    return html;
}

//4) Carrega e renderiza tudo
// MODIFICADO: Adicionado parâmetro isReadOnly
async function renderizarListItens(isReadOnly = false) {
    try {
        const data = (searchNome || searchCategoria)
            ? await buscarItens(searchNome, searchCategoria, currentPage, pageSize)
            : await carregarListItens(currentPage, pageSize);
        const categorias = await carregarCategorias();
    }

```

```

const categoryMap = {};
categorias.forEach(c => categoryMap[c.categoria_id] = c);

const main = document.getElementById('main-content');
if (main) {
  main.innerHTML = `
    ${criarSearchBar()}
    ${criarTabelaltens(data.items, categoryMap, isReadOnly)}
    ${criarControlesPaginacao(data.total_pages)}
  `;
  bindSearch();
  bindPagination(data.total_pages);
  if (!isReadOnly) { // Only bind row actions if not in read-only mode
    bindRowActions(categorias);
  }
} else {
  console.error("Elemento 'main-content' não encontrado em listar-itens.js");
}
} catch (err) {
  console.error(err);
  const main = document.getElementById('main-content');
  if (main) {
    main.innerHTML = `<div class="alert alert-warning">Erro: ${err.message}</div>`;
  }
}
}

//5) Bindings de busca
function bindSearch() {
  document.getElementById('btn-search')?.addEventListener('click', e => {
    e.preventDefault();
    const nome = document.getElementById('search-nome').value.trim();
    const cat = document.getElementById('search-categoria').value.trim();
    searchNome = nome;
    searchCategoria = cat;
    currentPage = 1;
    renderizarListItens();
  });

  document.getElementById('btn-clear-search')?.addEventListener('click', e => {
    e.preventDefault();
    searchNome = '';
    searchCategoria = '';
    currentPage = 1;
    pageSize = 10;
    renderizarListItens();
  });
}

//6) Bindings de paginação
function bindPagination(totalPages) {
  const itensPaginationNav = document.getElementById('itens-pagination-nav');
  const pageSizeSelect = document.getElementById('page-size-select');

  if (!itensPaginationNav) {
    console.warn("Elemento 'itens-pagination-nav' não encontrado para bindPagination.");
    return;
  }

  const handlePaginationClick = (e) => {
    e.preventDefault();
    const clickedPageLink = e.target.closest('a[data-page^="itens-"]');
    const clickedActionButton = e.target.closest('a[data-action^="itens-"]');
  }
}

```

```

    if (clickedPageLink) {
      const pageValue = clickedPageLink.dataset.page.split('-')[1];
      const newPage = parseInt(pageValue);
      if (!isNaN(newPage) && newPage !== currentPage) {
        currentPage = newPage;
        renderizarListItens();
      }
      return;
    }

    if (clickedActionButton) {
      const action = clickedActionButton.dataset.action;
      let newPage = currentPage;

      if (action === 'itens-prev') {
        if (newPage > 1) newPage--;
      } else if (action === 'itens-next') {
        if (newPage < totalPages) newPage++;
      }

      if (newPage !== currentPage) {
        currentPage = newPage;
        renderizarListItens();
      }
      return;
    }
  }
};

const handlePageSizeChange = (e) => {
  pageSize = parseInt(e.target.value);
  currentPage = 1;
  renderizarListItens();
};

itensPaginationNav.removeEventListener('click', handlePaginationClick);
if (pageSizeSelect) {
  pageSizeSelect.removeEventListener('change', handlePageSizeChange);
}

itensPaginationNav.addEventListener('click', handlePaginationClick);
if (pageSizeSelect) {
  pageSizeSelect.addEventListener('change', handlePageSizeChange);
}
}

//7) Bindings de ações nas linhas
function bindRowActions(categorias) {
  // deletar
  document.querySelectorAll('.btn-deletar').forEach(btn => {
    btn.onclick = async () => {
      if (!confirm('Excluir este item?')) return;

      const id = btn.dataset.id;
      const token = localStorage.getItem('token');
      try {
        const resp = await fetch(`/api/almoxarifado/itens/${id}`, {
          method: 'DELETE',
          headers: { 'Authorization': `Bearer ${token}` }
        });

        if (!resp.ok) {
          const error = await resp.json();
          const matches = error.detail?.match(/retirada_item\\.DETALH: Chave \\(item_id\\)=\\((\\d+

```

```

        if (matches && matches[1]) {
            throw new Error(`Item vinculado à retirada ID: ${matches[1]}. Não pode ser exclu
        }
        throw new Error(error.detail || 'Erro ao excluir item');
    }
    renderizarListItens();
} catch (err) {
    console.error(err);
    if (err.message.includes('retirada_item')) {
        alert('Item vinculado a uma ou mais retiradas.\n\nPrimeiro exclua as retiradas relac
    } else {
        alert(err.message);
    }
}
    };
});

// editar
document.querySelectorAll('.btn-editar').forEach(btn => {
    btn.onclick = async e => {
        e.preventDefault();
        const id = btn.dataset.id;
        const token = localStorage.getItem('token');

        //1) busca item
        const respItem = await fetch(`/api/almoxarifado/itens/${id}`, {
            headers: { 'Authorization': `Bearer ${token}` }
        });
        if (!respItem.ok) return alert('Erro ao carregar item');
        const item = await respItem.json();

        // 2) popula categorias no select de edição
        const sel = document.getElementById('edit-categoria_id');
        sel.innerHTML = `<option value="" disabled>Carregando...</option>`;
        categorias.forEach(c => {
            const o = document.createElement('option');
            o.value = c.categoria_id;
            o.textContent = `${c.categoria_id} ${c.nome_original.toUpperCase()}`;
            sel.append(o);
        });
        sel.value = item.categoria_id;

        //3) preenche demais campos
        const form = document.getElementById('form-editar-item');
        form.nome_item.value = item.nome_item_original;
        form.unidade_medida_item.value = item.unidade_medida_item;
        form.descricao_item.value = item.descricao_item;
        form.quantidade_item.value = item.quantidade_item;
        form.quantidade_minima_item.value = item.quantidade_minima_item || '';
        form.data_validade_item.value = item.data_validade_item?.split('T')[0] || '';
        form.data_entrada_item.value = item.data_entrada_item?.slice(0, 16) || '';
        form.marca_item.value = item.marca_item || '';

        //4) guarda id no botão salvar
        const saveBtn = document.getElementById('btn-salvar-editar-item');
        saveBtn.dataset.id = id;

        //5) exibe modal
        new bootstrap.Modal(
            document.getElementById('modalEditarItem')
        ).show();
    };
});

```

```

}

//8) salvar edição
document.getElementById('btn-salvar-editar-item')?.addEventListener('click', async e => {
  const id = e.currentTarget.dataset.id;
  if (!id) return alert('ID de edição não definido');

  const form = document.getElementById('form-editar-item');
  if (!form.checkValidity()) {
    form.reportValidity();
    return;
  }

  const data = {
    nome_item: form.nome_item.value.trim(),
    unidade_medida_item: form.unidade_medida_item.value.trim(),
    descricao_item: form.descricao_item.value.trim(),
    quantidade_item: Number(form.quantidade_item.value),
    categoria_id: Number(form.categoria_id.value),
  };

  // Campos opcionais
  if (form.quantidade_minima_item.value.trim())
    data.quantidade_minima_item = Number(form.quantidade_minima_item.value);
  if (form.data_validade_item.value.trim())
    data.data_validade_item = form.data_validade_item.value;
  if (form.data_entrada_item.value.trim())
    data.data_entrada_item = form.data_entrada_item.value;
  if (form.marca_item.value.trim())
    data.marca_item = form.marca_item.value.trim();

  try {
    const token = localStorage.getItem('token');
    const resp = await fetch(`/api/almoxarifado/itens/${id}`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`
      },
      body: JSON.stringify(data)
    });

    if (!resp.ok) {
      const err = await resp.json();
      throw new Error(err.detail || `Erro HTTP ${resp.status}`);
    }

    bootstrap.Modal.getInstance(document.getElementById('modalEditarItem')).hide();
    renderizarListItens();
  } catch (err) {
    console.error(err);
    alert('Erro ao salvar: ' + err.message);
  }
});

// 9) inicialização
const linkListar = document.getElementById('listar-item-link');
if (linkListar) {
  linkListar.addEventListener('click', e => {
    e.preventDefault();
    currentPage = 1;
    pageSize = 10;
    renderizarListItens();
  });
}

```

```

    });
}

// EXPOR A FUNÇÃO GLOBALMENTE
window.renderizarListItens = renderizarListItens;

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\login.js

```

//frontend\static\js\login.js

window.addEventListener('DOMContentLoaded', () => {
  const form = document.getElementById('login-form');
  if (!form) return;

  form.addEventListener('submit', async (e) => {
    e.preventDefault();

    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;
    const body = new URLSearchParams({ username, password });

    try {
      const resp = await fetch('api/almoxarifado/usuarios/token', {
        method: 'POST',
        headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
        credentials: 'include',
        body: body.toString()
      });

      if (!resp.ok) {
        const err = await resp.json();
        alert(err.detail || 'Falha ao autenticar');
        return;
      }

      const { access_token } = await resp.json();

      // ■■■■ AQUI: grava o JWT no localStorage para uso futuro ■■■■
      localStorage.setItem('token', access_token);

      const payload = JSON.parse(atob(access_token.split('.')[1]));
      const tipo = payload.tipo_usuario;

      if (tipo === 1) window.location.href = '/dashboardServidor';
      else if (tipo === 2) window.location.href = '/dashboardAlmoxarifado';
      else if (tipo === 3) window.location.href = '/dashboardDirecao';
      else alert('Tipo de usuário desconhecido.');
```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\logout.js

```
//frontend\static\js\logout.js

window.addEventListener('DOMContentLoaded', () => {
  const logoutLink = document.getElementById('logout-link');
  if (!logoutLink) return;

  logoutLink.addEventListener('click', async (e) => {
    e.preventDefault();

    // Tenta chamar o endpoint (opcional)
    try {
      await fetch('/api/almoxarifado/usuarios/logout', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        credentials: 'include'
      });
    } catch (err) {
      console.warn('Logout no servidor falhou, mas continua no client', err);
    }

    // Limpa o token e volta para a tela de login
    localStorage.removeItem('token');
    window.location.href = '/'; // ou '/index.html'
  });
});
```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\main.js

```
// frontend/static/js/main.js

import { retiradasModule } from './retiradasModule.js';
import { solicitarRetiradaModule } from './solicitar-retirada.js';
import { selecionarItemModule } from './selecionar-item-module.js';
import { reportsModule } from './reportsModule.js';
import { alertasModule } from './alertasModule.js';
import { apiService } from './apiService.js';
import { setNewAlertsFlag, getNewAlertsFlag, updateNotificationBellUI, showAlert, setNewWithdrawalRequest } from './uiService.js';
import estadoGlobal from './estadoGlobal.js';
import { dataService } from './dataService.js';
import { historicoServidorModule } from './historicoServidorModule.js';

const NOTIFICATION_SOUND_PATH = '/static/audio/notificacao01.mp3';
const NOTIFICATION_SOUND_PATH_RETIRADA = '/static/audio/notificacao02.mp3';

// Função auxiliar para re-inicializar dropdowns do Bootstrap
export function reinitializeBootstrapDropdowns() {
  const dropdownToggleList = [].slice.call(document.querySelectorAll('[data-bs-toggle="dropdown"]'));
  dropdownToggleList.map(function (dropdownToggleEl) {
    const dropdownInstance = bootstrap.Dropdown.getInstance(dropdownToggleEl);
    if (dropdownInstance) {
      return dropdownInstance;
    }
    return new bootstrap.Dropdown(dropdownToggleEl);
  });
}

const mainContent = document.getElementById('main-content');
let defaultHTML = mainContent ? mainContent.innerHTML : ''; // Armazena o conteúdo inicial
document.addEventListener('DOMContentLoaded', () => {
```



```

// Determine the dashboard type once on DOMContentLoaded
const currentPath = window.location.pathname;
const isServidorDashboard = currentPath.includes('/dashboardServidor');
const isAlmoxarifadoDashboard = currentPath.includes('/dashboardAlmoxarifado'); // Adicionado para A

const homeButton = document.getElementById('home-button');
if (homeButton && mainContent) {
    homeButton.addEventListener('click', e => {
        e.preventDefault();
        mainContent.innerHTML = defaultHTML;
        // Chamar a função de overview correta se for o dashboard
        if (isServidorDashboard) {
            window.loadDashboardOverview();
        } else if (isAlmoxarifadoDashboard) { // Adicionado para Almoxarifado
            loadAlmoxarifadoWelcomeSection();
        }
        bindQuickAccessLinks();
        bindLogoutLink();
        checkAlertsNotification(); // (Manter para a verificação inicial ao carregar a página)
        reinitializeBootstrapDropdowns();
    });
}

function bindQuickAccessLinks() {
    // Vincula o clique do card "Solicitar Retirada de Itens"
    document.getElementById('solicitar-retirada-servidor-link')?.addEventListener('click', e => {
        e.preventDefault();
        solicitarRetiradaModule.openModal(isServidorDashboard); // Passa a flag isServidorDashboard
    });

    // Vincula o clique do card "Meu Histórico de Retiradas"
    document.getElementById('historico-retiradas-servidor-link')?.addEventListener('click', e => {
        e.preventDefault();
        historicoServidorModule.renderMinhasRetiradas();
    });

    const modalCadastrarItemEl = document.getElementById('modalCadastrarItem');
    if (modalCadastrarItemEl) {
        const modalItem = new bootstrap.Modal(modalCadastrarItemEl);
        document.querySelectorAll('#btn-open-cadastrar-item').forEach(btn => {
            btn.onclick = e => {
                e.preventDefault();
                modalItem.show();
            };
        });
    }

    document.getElementById('listar-item-link')?.addEventListener('click', e => {
        e.preventDefault();
        if (typeof window.renderizarListItens === 'function') window.renderizarListItens();
        else console.warn("Função global 'renderizarListItens' não encontrada. Verifique se listar-i");
    });

    document.getElementById('listar-item-link-quick')?.addEventListener('click', e => {
        e.preventDefault();
        if (typeof window.renderizarListItens === 'function') window.renderizarListItens();
        else console.warn("Função global 'renderizarListItens' não encontrada.");
    });

    const modalCadastrarCategoriaEl = document.getElementById('modalCadastrarCategoria');
    if (modalCadastrarCategoriaEl) {
        const modalcat = new bootstrap.Modal(modalCadastrarCategoriaEl);
        document.getElementById('btn-open-cadastrar-categoria')?.addEventListener('click', e => {

```

```

        e.preventDefault();
        modalcat.show();
    });
}

document.getElementById('listar-categoria-link')?.addEventListener('click', e => {
    e.preventDefault();
    if (typeof window.renderizarCategorias === 'function') window.renderizarCategorias();
    else console.warn("Função global 'renderizarCategorias' não encontrada. Verifique se listar-");
});

document.getElementById('listar-categoria-link-quick')?.addEventListener('click', e => {
    e.preventDefault();
    if (typeof window.renderizarCategorias === 'function') window.renderizarCategorias();
    else console.warn("Função global 'renderizarCategorias' não encontrada.");
});

document.getElementById('listar-retiradas-pendentes-quick')?.addEventListener('click', e => {
    e.preventDefault();
    retiradasModule.renderPendentesRetiradas();
});

// Modal de relatórios
document.getElementById('open-reports-dashboard')?.addEventListener('click', e => {
    e.preventDefault();
    reportsModule.modalReportsDashboard.show();
});

// Listener para o link "Listar Alertas" do menu de navegação principal
document.getElementById('open-alertas-modal')?.addEventListener('click', e => {
    e.preventDefault();
    alertasModule.renderAlertsPage();
});

// Clique no sino de notificação
const alertBell = document.getElementById('alert-notification-bell');
if (alertBell) {
    // O sino é um dropdown toggle, seu comportamento é gerido pelo Bootstrap
    // Os itens do dropdown terão seus próprios listeners
}

const newWithdrawalRequestsMenuItem = document.getElementById('new-withdrawal-requests-menu-item');
const newAlertsMenuItem = document.getElementById('new-alerts-menu-item');
const openAllNotificationsLink = document.getElementById('open-all-notifications-link');

// Função para fechar o dropdown do sino
const hideNotificationDropdown = () => {
    bootstrap.Dropdown.getInstance(document.getElementById('alert-notification-bell'))?.hide();
};

// Comportamento para "Novos Alertas"
if (newAlertsMenuItem) {
    newAlertsMenuItem.addEventListener('click', e => {
        e.preventDefault();
        setNewAlertsFlag(false);
        updateNotificationBellUI();
        if (isServidorDashboard) {
            historicoServidorModule.renderMinhasRetiradas();
        } else {
            alertasModule.renderAlertsPage();
        }
        hideNotificationDropdown();
    });
}

```

```

}

// Comportamento para "Novas Solicitações de Retirada"
if (newWithdrawalRequestsMenuItem) {
  newWithdrawalRequestsMenuItem.addEventListener('click', e => {
    e.preventDefault();
    setNewWithdrawalRequestsFlag(false);
    updateNotificationBellUI();
    if (isServidorDashboard) {
      historicoServidorModule.renderMinhasRetiradas();
    } else {
      retiradasModule.renderPendentesRetiradas();
    }
    hideNotificationDropdown();
  });
}

// Adiciona listener para o link "Importar Tabela"
document.getElementById('btn-open-importar-tabela')?.addEventListener('click', e => {
  e.preventDefault();
  try {
    console.log('uiService no click do botão', uiService);
    const modalImportarTabela = uiService.getModalInstance('modalImportarTabelaltens');

    document.getElementById('form-importar-tabela-itens').reset();
    document.getElementById('import-feedback').style.display = 'none';
    document.getElementById('import-alert').className = 'alert';
    document.getElementById('import-alert').textContent = '';
    document.getElementById('import-errors-list').innerHTML = '';

    modalImportarTabela.show();
  } catch (error) {
    console.error("Erro ao abrir modal de importação:", error);
    showAlert("Erro ao tentar abrir a tela de importação. Por favor, tente novamente.", "dan");
  }
});

// NOVO: Adiciona listener para o botão de enviar tabela
document.getElementById('btn-enviar-tabela-itens')?.addEventListener('click', async () => {
  const form = document.getElementById('form-importar-tabela-itens');
  const fileInput = document.getElementById('arquivoltens');
  const importFeedback = document.getElementById('import-feedback');
  const importAlert = document.getElementById('import-alert');
  const importErrorsList = document.getElementById('import-errors-list');

  if (!form.checkValidity()) {
    form.reportValidity();
    return;
  }

  if (!fileInput.files || fileInput.files.length === 0) {
    showAlert('Por favor, selecione um arquivo para enviar.', 'warning');
    return;
  }

  const file = fileInput.files[0];

  uiService.showLoading();
  importFeedback.style.display = 'block';
  importAlert.className = 'alert';
  importAlert.textContent = '';
  importErrorsList.innerHTML = '';
  try {

```

```

const result = await apiService.uploadBulkItems(file);

importAlert.classList.add('alert-success');
importAlert.textContent = `Processamento concluído: ${result.total_items_processed} itens`;

if (result.errors && result.errors.length > 0) {
  importAlert.classList.remove('alert-success');
  importAlert.classList.add('alert-warning');
  importAlert.textContent += ` Foram encontrados ${result.errors.length} erros.`;
  result.errors.forEach(error => {
    const li = document.createElement('li');
    li.className = 'list-group-item list-group-item-danger';
    li.textContent = `Linha ${error.row}: ${error.error}`;
    importErrorsList.appendChild(li);
  });
}

if (typeof window.renderizarListItens === 'function') {
  window.renderizarListItens();
}

} catch (error) {
  importAlert.classList.add('alert-danger');
  importAlert.textContent = `Erro ao importar tabela: ${error.message || 'Erro desconhecido'}`;
  console.error('Erro no upload em massa', error);
} finally {
  uiService.hideLoading();
}
});

// Handler para o link "Solicitar Retirada" no menu de navegação (para Almoxarifado/Direcao)
document.getElementById('btn-open-solicitar-retirada')?.addEventListener('click', e => {
  e.preventDefault();
  solicitarRetiradaModule.openModal(false); // Sempre false para este link (não é dashboard do
});
}

function bindLogoutLink() {
  document.getElementById('logout-link')?.addEventListener('click', e => {
    e.preventDefault();
    if (typeof window.logout === 'function') window.logout();
    else console.warn("Função global 'logout' não encontrada. Verifique se logout.js está carregada.");
  });
}

async function checkAlertsNotification() {
  try {
    const alertsCount = await apiService.getUnviewedAlertsCount();

    if (alertsCount > 0) {
      setNewAlertsFlag(true);
    } else {
      setNewAlertsFlag(false);
    }
    updateNotificationBellUI();
  } catch (error) {
    console.error('checkAlertsNotification: Erro ao verificar notificações', error);
    setNewAlertsFlag(false);
    setNewWithdrawalRequestsFlag(false);
  }
}

let ws;

```

```

function connectAlertsWebSocket() {
  const userId = getUserIdFromToken();
  console.log("UserID para WebSocket:", userId);

  const protocol = window.location.protocol === 'https:' ? 'wss:' : 'ws:';
  const wsUrl = `${protocol}://${window.location.host}/api/almoxarifado/ws/alerts${userId ? `?user=${userId}` : ''}`;

  ws = new WebSocket(wsUrl);

  ws.onopen = (event) => {
    console.log("WebSocket para notificações conectado:", event);
  };

  ws.onmessage = (event) => {
    const message = JSON.parse(event.data);
    console.log("Mensagem WebSocket recebida", message);

    const isServidorDashboard = window.location.pathname.includes('/dashboardServidor');

    if (isServidorDashboard) {
      if (message.type === "withdrawal_status_update") {
        setNewWithdrawalRequestsFlag(true);
        const statusText = estadoGlobal.statusMap[message.status] || 'Desconhecido';
        showAlert(`Sua solicitação de retirada ID ${message.retirada_id} foi atualizada para ${statusText}`);
        try {
          const audio = new Audio(NOTIFICATION_SOUND_PATH_RETIRADA);
          audio.play().catch(e => console.error("Erro ao tocar som de notificação de retirada:", e));
        } catch (e) {
          console.error("Não foi possível criar objeto de audio para notificação:", e);
        }
        window.loadDashboardOverview(); // Chamar a função global
      }
    } else { // Para Almoxarifado/Direção
      if (message.type === "new_alert") {
        setNewAlertsFlag(true);
        showAlert("Novo alerta: " + message.message, "info", 5000);
        try {
          const audio = new Audio(NOTIFICATION_SOUND_PATH);
          audio.play().catch(e => console.error("Erro ao tocar som de notificação:", e));
        } catch (e) {
          console.error("Não foi possível criar objeto de áudio para notificação:", e);
        }
      } else if (message.type === "new_withdrawal_request") {
        setNewWithdrawalRequestsFlag(true);
        showAlert("Nova solicitação de retirada: " + message.message, "primary", 5000);
        try {
          const audio = new Audio(NOTIFICATION_SOUND_PATH_RETIRADA);
          audio.play().catch(e => console.error("Erro ao tocar som de notificação:", e));
        } catch (e) {
          console.error("Não foi possível criar objeto de áudio para notificação:", e);
        }
      }
    }
  }

  updateNotificationBellUI();
};

ws.onclose = (event) => {
  console.warn("WebSocket para notificações desconectado:", event.code, event.reason);
  setTimeout(connectAlertsWebSocket, 5000);
};

ws.onerror = (error) => {
  console.error("Erro no WebSocket para notificações:", error);
};

```

```

        ws.close();
    };
}

connectAlertsWebSocket();
reinitializeBootstrapDropdowns();

// Exportar loadDashboardOverview para ser acessível globalmente (Servidor)
window.loadDashboardOverview = async function() {
    uiService.showLoading();
    const loadingRecentWithdrawals = document.getElementById('loading-recent-withdrawals');
    const noRecentWithdrawals = document.getElementById('no-recent-withdrawals');

    try {
        const userId = getUserIdFromToken();
        if (userId) {
            const welcomeUserName = document.getElementById('welcome-user-name');
            const userSiape = document.getElementById('user-siape');
            const userSector = document.getElementById('user-sector');

            if (welcomeUserName && userSiape && userSector) {
                const userDetails = await apiService.getCurrentUserDetails(userId);
                welcomeUserName.textContent = userDetails.name;
                userSiape.textContent = userDetails.siape || 'N/D';
                userSector.textContent = userDetails.sectorName || 'N/D';
            } else {
                console.warn("Elementos de detalhes do usuário não encontrados no DOM. Ignorando atualizações.");
            }

            if (loadingRecentWithdrawals) loadingRecentWithdrawals.style.display = 'block';
            if (noRecentWithdrawals) noRecentWithdrawals.style.display = 'none';

            const recentWithdrawalsData = await dataService.getProcessedRetiradas(apiService.fetchUserWithdrawals);

            estadoGlobal.minhasRetiradas = recentWithdrawalsData.items;
            renderRecentWithdrawals(recentWithdrawalsData.items);
        } else {
            console.warn("Usuário não logado ou ID de usuário não encontrado para carregar o overview");
            if (loadingRecentWithdrawals) loadingRecentWithdrawals.style.display = 'none';
            if (noRecentWithdrawals) noRecentWithdrawals.style.display = 'block';
        }
    } catch (error) {
        console.error("Erro ao carregar o overview do dashboard do servidor:", error);
        showAlert("Erro ao carregar informações do dashboard. Tente novamente.", "danger");
        if (loadingRecentWithdrawals) loadingRecentWithdrawals.style.display = 'none';
        if (noRecentWithdrawals) {
            noRecentWithdrawals.textContent = "Erro ao carregar solicitações.";
            noRecentWithdrawals.style.display = 'block';
        }
    } finally {
        uiService.hideLoading();
    }
}

// Função para carregar e exibir as informações do usuário no dashboard do Almoxarifado
async function loadAlmoxarifadoWelcomeSection() {
    uiService.showLoading();
    const welcomeUserNameAlmoxarifado = document.getElementById('welcome-user-name-almoxarifado');
    const userSiapeAlmoxarifado = document.getElementById('user-siape-almoxarifado');
    const userSectorAlmoxarifado = document.getElementById('user-sector-almoxarifado');

    try {
        const userId = getUserIdFromToken();

```

```

    if (userId) {
      if (welcomeUserNameAlmoxarifado && userSiapeAlmoxarifado && userSectorAlmoxarifado) {
        const userDetails = await apiService.getCurrentUserDetails(userId);
        welcomeUserNameAlmoxarifado.textContent = userDetails.name || 'Usuário';
        userSiapeAlmoxarifado.textContent = userDetails.siape || 'N/D';
        userSectorAlmoxarifado.textContent = userDetails.sectorName || 'N/D';
      } else {
        console.warn("Elementos de detalhes do usuário para Almoxarifado não encontrados no");
      }
    } else {
      console.warn("Usuário não logado ou ID de usuário não encontrado para carregar o overview");
      if (welcomeUserNameAlmoxarifado) welcomeUserNameAlmoxarifado.textContent = 'Usuário';
      if (userSiapeAlmoxarifado) userSiapeAlmoxarifado.textContent = 'N/D';
      if (userSectorAlmoxarifado) userSectorAlmoxarifado.textContent = 'N/D';
    }
  } catch (error) {
    console.error("Erro ao carregar o overview do dashboard do almoxarifado:", error);
    showAlert("Erro ao carregar informações do dashboard do almoxarifado. Tente novamente.", "da");
    if (welcomeUserNameAlmoxarifado) welcomeUserNameAlmoxarifado.textContent = 'Erro';
    if (userSiapeAlmoxarifado) userSiapeAlmoxarifado.textContent = 'N/D';
    if (userSectorAlmoxarifado) userSectorAlmoxarifado.textContent = 'N/D';
  } finally {
    uiService.hideLoading();
  }
}

```

```

function renderRecentWithdrawals(withdrawals) {
  const container = document.getElementById('latest-withdrawals-container');
  if (!container) {
    console.warn("Elemento 'latest-withdrawals-container' não encontrado. Não é um dashboard de");
    return;
  }

  const loadingRecentWithdrawals = document.getElementById('loading-recent-withdrawals');
  const noRecentWithdrawals = document.getElementById('no-recent-withdrawals');

  if (loadingRecentWithdrawals) loadingRecentWithdrawals.style.display = 'none';
  if (noRecentWithdrawals) noRecentWithdrawals.style.display = 'none';

  container.innerHTML = ''; // Limpa o conteúdo existente

  if (withdrawals.length === 0) {
    if (noRecentWithdrawals) noRecentWithdrawals.style.display = 'block';
    return;
  }

  withdrawals.forEach(retirada => {
    let progress = 0;
    let progressBarClass = 'bg-secondary';
    let statusText = estadoGlobal.statusMap[retirada.status] || 'Desconhecido';

    switch (retirada.status) {
      case 1:
        progress = 25;
        progressBarClass = 'bg-info';
        break;
      case 2:
        progress = 50;
        progressBarClass = 'bg-primary';
        break;
      case 3:
        progress = 100;

```

```

        progressBarClass = 'bg-success';
        break;
    case 4:
        progress = 100;
        progressBarClass = 'bg-danger';
        break;
    }

    const withdrawalHtml = `
        <div class="list-group-item list-group-item-action py-3 px-4 mb-2 rounded-lg shadow-sm r
            <div class="d-flex w-100 justify-content-between align-items-center">
                <h6 class="mb-1">Solicitação ID: ${retirada.retirada_id}</h6>
                <small class="text-muted">${formatDateTime(retirada.data_solicitacao)}</small>
            </div>
            <p class="mb-2">Status: <strong>${statusText}</strong></p>
            <div class="progress" style="height: 10px;">
                <div class="progress-bar ${progressBarClass}" role="progressbar" style="width: $
            </div>
            <small class="text-muted mt-2 d-block">
                ${retirada.justificativa ? `Justificativa: ${retirada.justificativa}` : ''}
                ${retirada.detalhe_status ? `Detalhe: ${retirada.detalhe_status}` : ''}
            </small>
        </div>
    `;
    container.insertAdjacentHTML('beforeend', withdrawalHtml);
});

container.removeEventListener('click', handleRecentWithdrawalItemClick);
container.addEventListener('click', handleRecentWithdrawalItemClick);
}

function handleRecentWithdrawalItemClick(e) {
    const clickedItem = e.target.closest('.recent-withdrawal-item');
    if (clickedItem) {
        const retiradaId = parseInt(clickedItem.dataset.retiradaId);
        const retirada = estadoGlobal.minhasRetiradas.find(r => r.retirada_id === retiradaId);

        if (retirada) {
            uiService.fillModalDetalhes(retirada, true);
            uiService.getModalInstance('modalVerDetalhesRetirada').show();
        } else {
            uiService.showLoading();
            apiService.get(`/retiradas/${retiradaId}`)
                .then(rawRetirada => dataService.getProcessedRetiradas(async () => ({ items: [rawRet
                .then(processedData => {
                    const fullRetirada = processedData.items[0];
                    uiService.fillModalDetalhes(fullRetirada, true);
                    uiService.getModalInstance('modalVerDetalhesRetirada').show();
                })
                .catch(error => {
                    console.error("Erro ao carregar detalhes da solicitação:", error);
                    showAlert('Não foi possível carregar os detalhes da solicitação. Tente novamente
                })
                .finally(() => {
                    uiService.hideLoading();
                });
            });
        }
    }
}

bindLogoutLink();
document.getElementById('listar-retiradas-link')?.addEventListener('click', e => {

```



```

        e.preventDefault();
        retiradasModule.renderHistoricoRetiradas();
    });

    document.getElementById('listar-retiradas-pendentes-link')?.addEventListener('click', e => {
        e.preventDefault();
        retiradasModule.renderPendentesRetiradas();
    });

    const btnConfirmarAutorizar = document.getElementById('btn-confirmar-autorizar-retirada');
    if (btnConfirmarAutorizar) {
        btnConfirmarAutorizar.addEventListener('click', retiradasModule._handleAuthorizeDeny.bind(retiradasModule));
    }

    const btnConfirmarNegar = document.getElementById('btn-confirmar-negar-retirada');
    if (btnConfirmarNegar) {
        btnConfirmarNegar.addEventListener('click', retiradasModule._handleAuthorizeDeny.bind(retiradasModule));
    }

    mainContent.addEventListener('click', e => {
        const viewAllLink = e.target.closest('#view-all-my-withdrawals-link');
        if (viewAllLink) {
            e.preventDefault();
            historicoServidorModule.renderMinhasRetiradas();
        }
    });

    solicitarRetiradaModule.init();
    selecionarItemModule.init();
    reportsModule.init();
    alertasModule.init();
    historicoServidorModule.init();

    bindQuickAccessLinks();

    checkAlertsNotification();

    // Inicializa o dashboard correto ao carregar a página
    if (isServidorDashboard) {
        window.loadDashboardOverview();
    } else if (isAlmoxarifadoDashboard) {
        loadAlmoxarifadoWelcomeSection();
    }
});

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\main_direcao.js

```

// frontend/static/js/main_direcao.js

import { retiradasModule } from './retiradasModule.js';
import { solicitarRetiradaModule } from './solicitar-retirada.js';
import { selecionarItemModule } from './selecionar-item-module.js';
import { reportsModule } from './reportsModule.js';
import { alertasModule } from './alertasModule.js';
import { apiService } from './apiService.js';
import { setNewAlertsFlag, updateNotificationBellUI, showAlert, setNewWithdrawalRequestsFlag, getUserId } from './apiService.js';
import { usuariosModule } from './usuariosModule.js';
import { setoresModule } from './setoresModule.js';
import { uiService } from './uiService.js';

```

```

const NOTIFICATION_SOUND_PATH = '/static/audio/notificacao01.mp3';

// Função auxiliar para re-inicializar dropdowns do Bootstrap
export function reinitializeBootstrapDropdowns() {
  const dropdownToggleList = [].slice.call(document.querySelectorAll('[data-bs-toggle="dropdown"]'));
  dropdownToggleList.map(function (dropdownToggleEl) {
    const dropdownInstance = bootstrap.Dropdown.getInstance(dropdownToggleEl);
    if (dropdownInstance) {
      return new bootstrap.Dropdown(dropdownToggleEl);
    }
    return dropdownInstance;
  });
}

const mainContent = document.getElementById('main-content');
// defaultHTML deve ser capturado no DOMContentLoaded, mas apenas o HTML inicial do container
// e não o que pode ser injetado dinamicamente após
let defaultHTML = mainContent ? mainContent.innerHTML : '';

document.addEventListener('DOMContentLoaded', () => {
  const currentPath = window.location.pathname;
  const isDirecaoDashboard = currentPath.includes('/dashboardDirecao');

  const homeButton = document.getElementById('home-button');
  if (homeButton && mainContent) {
    homeButton.addEventListener('click', e => {
      e.preventDefault();
      // Restaura o HTML padrão do mainContent antes de qualquer outra operação
      // É importante que 'defaultHTML' contenha o estado original do div#main-content
      // para que todos os elementos sejam recriados e possam ter os listeners vinculados.
      mainContent.innerHTML = defaultHTML;

      // Rebind dos links e funções após restaurar o HTML
      bindDirecaoLinks();
      bindLogoutLink();
      checkAlertsNotification();
      reinitializeBootstrapDropdowns();

      // Chama a função de boas-vindas apenas se for o dashboard da Direção
      if (isDirecaoDashboard) {
        window.loadDirecaoWelcomeSection(); // Chama a função global
      }
    });
  }

  function bindDirecaoLinks() {
    // Usuários
    document.getElementById('listar-usuarios-link')?.addEventListener('click', e => {
      e.preventDefault();
      usuariosModule.renderUsuariosList();
    });
    // Adicionado para o quick access card
    document.getElementById('listar-usuarios-link-quick')?.addEventListener('click', e => {
      e.preventDefault();
      usuariosModule.renderUsuariosList();
    });

    document.getElementById('btn-open-cadastrar-usuario')?.addEventListener('click', e => {
      e.preventDefault();
      usuariosModule.modalCadastrarUsuario.show(); // Trigger modal directly
    });

    // Setores

```

```

document.getElementById('listar-setores-link')?.addEventListener('click', e => {
    e.preventDefault();
    setoresModule.renderSetoresList();
});
// Adicionado para o quick access card
document.getElementById('listar-setores-link-quick')?.addEventListener('click', e => {
    e.preventDefault();
    setoresModule.renderSetoresList();
});

document.getElementById('btn-open-cadastrar-setor')?.addEventListener('click', e => {
    e.preventDefault();
    setoresModule.modalCadastrarSetor.show();
});

// Itens
document.getElementById('listar-item-link')?.addEventListener('click', e => {
    e.preventDefault();
    if (typeof window.renderizarListItens === 'function') {
        window.renderizarListItens(true);
    } else {
        console.warn("Função global 'renderizarListItens' não encontrada.");
    }
});

// Retiradas
document.getElementById('btn-open-solicitar-retirada')?.addEventListener('click', e => {
    e.preventDefault();
    solicitarRetiradaModule.openModal(false);
});
document.getElementById('listar-retiradas-link')?.addEventListener('click', e => {
    e.preventDefault();
    retiradasModule.renderHistoricoRetiradas();
});
document.getElementById('listar-retiradas-pendentes-link')?.addEventListener('click', e => {
    e.preventDefault();
    retiradasModule.renderPendentesRetiradas();
});

// Relatórios
document.getElementById('open-reports-dashboard')?.addEventListener('click', e => {
    e.preventDefault();
    reportsModule.modalReportsDashboard.show();
});
// Adicionado para o quick access card
document.getElementById('open-reports-dashboard-quick')?.addEventListener('click', e => {
    e.preventDefault();
    reportsModule.modalReportsDashboard.show();
});

// Alertas
document.getElementById('open-alertas-modal')?.addEventListener('click', e => {
    e.preventDefault();
    alertasModule.renderAlertsPage();
});
// Adicionado para o quick access card
document.getElementById('open-alertas-modal-quick')?.addEventListener('click', e => {
    e.preventDefault();
    alertasModule.renderAlertsPage();
});

const newWithdrawalRequestsMenuItem = document.getElementById('new-withdrawal-requests-menu-item');
const newAlertsMenuItem = document.getElementById('new-alerts-menu-item');

```

```

const openAllNotificationsLink = document.getElementById('open-all-notifications-link');
const hideNotificationDropdown = () => {
    bootstrap.Dropdown.getInstance(document.getElementById('alert-notification-bell'))?.hide();
};

if (newAlertsMenuItem) {
    newAlertsMenuItem.addEventListener('click', e => {
        e.preventDefault();
        setNewAlertsFlag(false);
        updateNotificationBellUI();
        alertasModule.renderAlertsPage();
        hideNotificationDropdown();
    });
}

if (newWithdrawalRequestsMenuItem) {
    newWithdrawalRequestsMenuItem.addEventListener('click', e => {
        e.preventDefault();
        setNewWithdrawalRequestsFlag(false);
        updateNotificationBellUI();
        retiradasModule.renderPendentesRetiradas();
        hideNotificationDropdown();
    });
}

if (openAllNotificationsLink) {
    openAllNotificationsLink.addEventListener('click', e => {
        e.preventDefault();
        setNewAlertsFlag(false);
        setNewWithdrawalRequestsFlag(false);
        updateNotificationBellUI();
        alertasModule.renderAlertsPage();
        hideNotificationDropdown();
    });
}

function bindLogoutLink() {
    document.getElementById('logout-link')?.addEventListener('click', e => {
        e.preventDefault();
        if (typeof window.logout === 'function') window.logout();
        else console.warn("Função global 'logout' não encontrada.");
    });
}

async function checkAlertsNotification() {
    try {
        const alertsCount = await apiService.getUnviewedAlertsCount();
        if (alertsCount > 0) {
            setNewAlertsFlag(true);
        } else {
            setNewAlertsFlag(false);
        }
        updateNotificationBellUI();
    }
    catch (error) {
        console.error('checkAlertsNotification: Erro ao verificar notificações', error);
        setNewAlertsFlag(false);
        setNewWithdrawalRequestsFlag(false);
    }
}

let ws;

function connectAlertsWebSocket() {

```

```

const userId = getUserIdFromToken();
console.log("UserID para WebSocket:", userId);

const protocol = window.location.protocol === 'https:' ? 'wss:' : 'ws:';
const wsUrl = `${protocol}://${window.location.host}/api/almoxarifado/ws/alerts${userId ? `?user_` : ''}`;

ws = new WebSocket(wsUrl);

ws.onopen = (event) => {
  console.log("WebSocket para notificações conectado:", event);
};

ws.onmessage = (event) => {
  const message = JSON.parse(event.data);
  console.log("Mensagem WebSocket recebida", message);

  const isServidorDashboard = window.location.pathname.includes('/dashboardServidor');
  if (isServidorDashboard) {
    if (message.type === "withdrawal_status_update") {
      setNewWithdrawalRequestsFlag(true);
      const statusText = estadoGlobal.statusMap[message.status] || 'Desconhecido';
      showAlert(`Sua solicitação de retirada ID ${message.retirada_id} foi atualizada para ${statusText}`);
      try {
        const audio = new Audio(NOTIFICATION_SOUND_PATH_RETIRADA);
        audio.play().catch(e => console.error("Erro ao tocar som de notificação de retirada:", e));
      } catch (e) {
        console.error("Não foi possível criar objeto de audio para notificação:", e);
      }
      window.loadDashboardOverview(); // Chamar a função global
    }
  } else { // Para Almoxarifado/Direção
    if (message.type === "new_alert") {
      setNewAlertsFlag(true);
      showAlert("Novo alerta: " + message.message, "info", 5000);
      try {
        const audio = new Audio(NOTIFICATION_SOUND_PATH);
        audio.play().catch(e => console.error("Erro ao tocar som de notificação:", e));
      } catch (e) {
        console.error("Não foi possível criar objeto de áudio para notificação:", e);
      }
    } else if (message.type === "new_withdrawal_request") {
      setNewWithdrawalRequestsFlag(true);
      showAlert("Nova solicitação de retirada: " + message.message, "primary", 5000);
      try {
        const audio = new Audio(NOTIFICATION_SOUND_PATH_RETIRADA);
        audio.play().catch(e => console.error("Erro ao tocar som de notificação:", e));
      } catch (e) {
        console.error("Não foi possível criar objeto de áudio para notificação:", e);
      }
    }
  }
  updateNotificationBellUI();
};

ws.onclose = (event) => {
  console.warn("WebSocket para notificações desconectado:", event.code, event.reason);
  setTimeout(connectAlertsWebSocket, 5000);
};

ws.onerror = (error) => {
  console.error("Erro no WebSocket para notificações:", error);
  ws.close();
};

```

```

    }

    // Função para carregar e exibir as informações do usuário no dashboard da Direção
    // Esta função será exportada globalmente
    window.loadDirecaoWelcomeSection = async function() { // Tornada global
        uiService.showLoading();
        const welcomeUserNameDirecao = document.getElementById('welcome-user-name-direcao');
        const userSiapeDirecao = document.getElementById('user-siape-direcao');
        const userSectorDirecao = document.getElementById('user-sector-direcao');

        // Inicializa com texto padrão imediatamente para evitar "Carregando..." persistente
        if (welcomeUserNameDirecao) welcomeUserNameDirecao.textContent = 'Direção';
        if (userSiapeDirecao) userSiapeDirecao.textContent = 'N/D';
        if (userSectorDirecao) userSectorDirecao.textContent = 'N/D';

        try {
            const userId = getUserIdFromToken();
            if (userId) {
                if (welcomeUserNameDirecao && userSiapeDirecao && userSectorDirecao) {
                    const userDetails = await apiService.getCurrentUserDetails(userId);
                    welcomeUserNameDirecao.textContent = userDetails.name || 'Direção';
                    userSiapeDirecao.textContent = userDetails.siape || 'N/D';
                    userSectorDirecao.textContent = userDetails.sectorName || 'N/D';
                } else {
                    console.warn("Elementos de detalhes do usuário para Direção não encontrados no DOM.");
                }
            } else {
                console.warn("Usuário não logado ou ID de usuário não encontrado para carregar o overview");
            }
        } catch (error) {
            console.error("Erro ao carregar o overview do dashboard da Direção:", error);
            showAlert("Erro ao carregar informações do dashboard da Direção. Tente novamente.", "danger");
            if (welcomeUserNameDirecao) welcomeUserNameDirecao.textContent = 'Erro';
            if (userSiapeDirecao) userSiapeDirecao.textContent = 'N/D';
            if (userSectorDirecao) userSectorDirecao.textContent = 'N/D';
        } finally {
            uiService.hideLoading();
        }
    }

    // Inicializar modules
    solicitarRetiradaModule.init();
    selecionarItemModule.init();
    reportsModule.init();
    alertasModule.init();
    usuariosModule.init();
    setoresModule.init();

    // Os links de acesso rápido da direção e a seção de boas-vindas
    // são inicializados no final do DOMContentLoaded para garantir que o HTML esteja pronto.
    bindDirecaoLinks();
    checkAlertsNotification();
    connectAlertsWebSocket();
    reinitializeBootstrapDropdowns();

    // Chama a função de boas-vindas ao carregar o dashboard da Direção, mas apenas UMA VEZ
    // na carga inicial da página. Se o botão "Home" for clicado, ele chamará a função global.
    if (isDirecaoDashboard) {
        window.loadDirecaoWelcomeSection();
    }
});

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\reportsModule.js

```
// frontend/static/js/reportsModule.js

import { apiService } from './apiService.js';
import { uiService } from './uiService.js';
import { showAlert } from './utils.js';
import { selecionarItemModule } from './selecionar-item-module.js';

class ReportsModule {
  constructor() {
    // References to main report dashboard modal
    this.modalReportsDashboard = uiService.getModalInstance('modalReportsDashboard');

    // References for Quantidade de Itens report modal
    this.modalReportQuantidadeItens = uiService.getModalInstance('modalReportQuantidadeItens');
    this.formReportQuantidadeItens = document.getElementById('form-report-quantidade-itens');
    this.btnGerarReportQuantidadeItens = document.getElementById('btnGerarReportQuantidadeItens');
    this.filterCategoryReportSelect = document.getElementById('filterCategoryReport');

    // Novas referências para seleção de produto
    this.selectedProductNameReportInput = document.getElementById('selectedProductNameReport');
    this.selectedProductIdReportInput = document.getElementById('selectedProductIdReport'); // Campo
    this.btnOpenSelectProductReportModal = document.getElementById('btnOpenSelectProductReportModal');
    this.btnClearSelectedProductReport = document.getElementById('btnClearSelectedProductReport');

    // References for Entrada de Itens report modal
    this.modalReportEntradaItens = uiService.getModalInstance('modalReportEntradaItens');
    this.formReportEntradaItens = document.getElementById('form-report-entrada-itens');
    this.btnGerarReportEntradaItens = document.getElementById('btnGerarReportEntradaItens');

    // References for Retiradas por Setor report modal
    this.modalReportRetiradasSetor = uiService.getModalInstance('modalReportRetiradasSetor');
    this.formReportRetiradasSetor = document.getElementById('form-report-retiradas-setor');
    this.selectSetor = document.getElementById('selectSetor');
    this.btnGerarReportRetiradasSetor = document.getElementById('btnGerarReportRetiradasSetor');

    // References for Retiradas por Usuário report modal
    this.modalReportRetiradasUsuario = uiService.getModalInstance('modalReportRetiradasUsuario');
    this.formReportRetiradasUsuario = document.getElementById('form-report-retiradas-usuario');
    this.selectUsuario = document.getElementById('selectUsuario');
    this.btnGerarReportRetiradasUsuario = document.getElementById('btnGerarReportRetiradasUsuario');
  }

  init() {
    this._bindEvents();
  }

  _bindEvents() {
    // --- Event listeners for generating reports ---
    if (this.btnGerarReportQuantidadeItens) {
      this.btnGerarReportQuantidadeItens.addEventListener('click', () => this._handleGerarReportQua
    }

    if (this.btnGerarReportEntradaItens) {
      this.btnGerarReportEntradaItens.addEventListener('click', () => this._handleGerarReportEntra
    }

    if (this.btnGerarReportRetiradasSetor) {
      this.btnGerarReportRetiradasSetor.addEventListener('click', () => this._handleGerarReportRet
    }
  }
}
```

```

    if (this.btnGerarReportRetiradasUsuario) {
        this.btnGerarReportRetiradasUsuario.addEventListener('click', () => this._handleGerarReportR
    }

    // --- Event listeners for modal show/hide to handle dynamic content and resets ---
    if (this.modalReportRetiradasSetor) {
        this.modalReportRetiradasSetor._element.addEventListener('show.bs.modal', () => this._popula
        this.modalReportRetiradasSetor._element.addEventListener('hidden.bs.modal', () => this.formR
    }
    if (this.modalReportRetiradasUsuario) {
        this.modalReportRetiradasUsuario._element.addEventListener('show.bs.modal', () => this._popu
        this.modalReportRetiradasUsuario._element.addEventListener('hidden.bs.modal', () => this.for
    }
    if (this.modalReportQuantidadeItens) {
        this.modalReportQuantidadeItens._element.addEventListener('show.bs.modal', () => this._popul
        this.modalReportQuantidadeItens._element.addEventListener('hidden.bs.modal', () => {
            this.formReportQuantidadeItens.reset();
            this._clearSelectedProductFilter();
        });

        // Eventos para o novo botão de seleção de produto e limpeza
        if (this.btnOpenSelectProductReportModal) {
            this.btnOpenSelectProductReportModal.addEventListener('click', () => {
                this.modalReportQuantidadeItens.hide();
                // Opcional: Adicione um listener temporário para reabrir o modal
                // quando o modal de seleção for escondido, independente de seleção.
                const reOpenReportModal = () => {
                    this.modalReportQuantidadeItens.show();
                    // Remove o listener temporário para evitar que ele reabra o modal em outros con
                    selecionarItemModule.modalSelecionarItem._element.removeEventListener('hidden.bs
                };
                selecionarItemModule.modalSelecionarItem._element.addEventListener('hidden.bs.modal'

                setTimeout(() => {
                    selecionarItemModule.openModal('itemSelectedForReportFilter', 10);
                }, 300);
            });
        }
        if (this.btnClearSelectedProductReport) {
            this.btnClearSelectedProductReport.addEventListener('click', () => this._clearSelectedPr
        }
    }
    if (this.modalReportEntradaItens) {
        this.modalReportEntradaItens._element.addEventListener('hidden.bs.modal', () => this.formRep
    }

    // Listener para o evento customizado disparado pelo selecionarItemModule (quando um item É sele
    document.addEventListener('itemSelectedForReportFilter', (event) => {
        this._handleItemSelectedForReportFilter(event.detail.item);
        // O modal de seleção já se fechou ao despachar o evento.
        // O listener 'hidden.bs.modal' adicionado acima já cuidará de reabrir o modal de relatório.
    });
}

async _populateSetores() {
    this.selectSetor.innerHTML = '<option value="" disabled selected>Carregando setores...</option>';
    uiService.showLoading();
    try {
        const setores = await apiService.fetchAllSetores();
        this.selectSetor.innerHTML = '<option value="" disabled selected>Selecione um setor</option>';
        setores.forEach(setor => {
            const option = document.createElement('option');
            option.value = setor.setor_id;

```



```

        option.textContent = setor.nome_setor;
        this.selectSetor.appendChild(option);
    });
} catch (error) {
    console.error('Erro ao carregar setores:', error);
    showAlert('Erro ao carregar setores para o relatório.', 'danger');
    this.selectSetor.innerHTML = '<option value="" disabled selected>Erro ao carregar</option>';
} finally {
    uiService.hideLoading();
}
}

async _populateUsuarios() {
    this.selectUsuario.innerHTML = '<option value="" disabled selected>Carregando usuários...</option>';
    uiService.showLoading();
    try {
        const usuarios = await apiService.get('/usuarios');
        this.selectUsuario.innerHTML = '<option value="" disabled selected>Selecione um usuário</option>';
        usuarios.forEach(usuario => {
            const option = document.createElement('option');
            option.value = usuario.usuario_id;
            option.textContent = usuario.nome_usuario;
            this.selectUsuario.appendChild(option);
        });
    } catch (error) {
        console.error('Erro ao carregar usuários:', error);
        showAlert('Erro ao carregar usuários para o relatório.', 'danger');
        this.selectUsuario.innerHTML = '<option value="" disabled selected>Erro ao carregar</option>';
    } finally {
        uiService.hideLoading();
    }
}

async _populateCategoriesForReport() {
    this.filterCategoryReportSelect.innerHTML = '<option value="">Todas as Categorias</option>';
    uiService.showLoading();
    try {
        const categorias = await apiService.get('/categorias');
        categorias.forEach(cat => {
            const option = document.createElement('option');
            option.value = cat.categoria_id;
            option.textContent = cat.nome_original;
            this.filterCategoryReportSelect.appendChild(option);
        });
    } catch (error) {
        console.error('Erro ao carregar categorias para o relatório:', error);
        showAlert('Erro ao carregar categorias.', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

_clearSelectedProductFilter() {
    this.selectedProductNameReportInput.value = '';
    this.selectedProductIdReportInput.value = '';
}

_handleItemSelectedForReportFilter(selectedItem) {
    if (selectedItem) {
        this.selectedProductNameReportInput.value = selectedItem.nome_item_original;
        this.selectedProductIdReportInput.value = selectedItem.item_id;
        showAlert('Produto "${selectedItem.nome_item_original}" selecionado para filtro.', 'info');
    } else {

```

```

        this._clearSelectedProductFilter();
        showAlert('Nenhum produto selecionado.', 'warning');
    }
}

async _handleGerarReportQuantidadeItens() {
    if (!this.formReportQuantidadeItens.checkValidity()) {
        this.formReportQuantidadeItens.reportValidity();
        return;
    }

    const filtro_categoria = this.filterCategoryReportSelect.value;
    const filtro_produto_id = this.selectedProductIdReportInput.value;
    const formato = document.getElementById('formatoQuantidadeItens').value;

    const params = new URLSearchParams({ formato });
    if (filtro_categoria) params.append('filtro_categoria', filtro_categoria);
    if (filtro_produto_id) {
        params.append('filtro_produto', this.selectedProductNameReportInput.value); // Envia o nome,
    }

    uiService.showLoading();
    try {
        const token = localStorage.getItem('token');
        const response = await fetch(`/api/almoxarifado/relatorios/quantidade-itens/?${params.toString()}`
            , {
                method: 'GET',
                headers: {
                    'Authorization': `Bearer ${token}`
                }
            });

        if (!response.ok) {
            const errorData = await response.json().catch(() => ({ detail: 'Erro desconhecido ao gerar relatório de quantidade de itens.' }));
            throw new Error(errorData.detail || `Erro HTTP: ${response.status}`);
        }

        const blob = await response.blob();
        const contentDisposition = response.headers.get('content-disposition');
        let filename = `relatorio_quantidade_itens.${formato}`;

        if (contentDisposition) {
            const filenameMatch = contentDisposition.match(/filename\*?=(?:UTF-8')?([^\;]+)/i);
            if (filenameMatch && filenameMatch[1]) {
                filename = decodeURIComponent(filenameMatch[1].replace(/"/g, ''));
            } else {
                const simpleSplit = contentDisposition.split('filename=')[1];
                if (simpleSplit) {
                    filename = simpleSplit.replace(/"/g, '');
                }
            }
        }

        this._downloadFile(blob, filename);

        showAlert('Relatório de quantidade de itens gerado com sucesso!', 'success');
        this.modalReportQuantidadeItens.hide();
    } catch (error) {
        console.error('Erro ao gerar relatório de quantidade de itens:', error);
        showAlert(error.message || 'Ocorreu um erro ao gerar o relatório de quantidade de itens.', 'error');
    } finally {
        uiService.hideLoading();
    }
}

```

```

async _handleGerarReportEntradaItens() {
  if (!this.formReportEntradaItens.checkValidity()) {
    this.formReportEntradaItens.reportValidity();
    return;
  }

  const data_inicio = document.getElementById('dataInicioEntrada').value;
  const data_fim = document.getElementById('dataFimEntrada').value;
  const formato = document.getElementById('formatoEntradaItens').value;

  if (!data_inicio || !data_fim) {
    showAlert('Por favor, preencha as datas inicial e final.', 'warning');
    return;
  }

  const params = new URLSearchParams({ data_inicio, data_fim, formato });

  uiService.showLoading();
  try {
    const token = localStorage.getItem('token');
    const response = await fetch(`/api/almoxarifado/relatorios/entrada-itens/?${params.toString()}`, {
      method: 'GET',
      headers: {
        'Authorization': `Bearer ${token}`
      }
    });

    if (!response.ok) {
      const errorData = await response.json().catch(() => ({ detail: 'Erro desconhecido ao gerar relatório de entrada de itens.' }));
      throw new Error(errorData.detail || `Erro HTTP: ${response.status}`);
    }

    const blob = await response.blob();
    const contentDisposition = response.headers.get('content-disposition');
    let filename = `relatorio_entrada_itens.${formato}`;

    if (contentDisposition) {
      const filenameMatch = contentDisposition.match(/filename\*?=(?:UTF-8'')?([^\s;]+)/i);
      if (filenameMatch && filenameMatch[1]) {
        filename = decodeURIComponent(filenameMatch[1].replace(/"/g, ''));
      } else {
        const simpleSplit = contentDisposition.split('filename=')[1];
        if (simpleSplit) {
          filename = simpleSplit.replace(/"/g, '');
        }
      }
    }

    this._downloadFile(blob, filename);

    showAlert('Relatório de entrada de itens gerado com sucesso!', 'success');
    this.modalReportEntradaItens.hide();
  } catch (error) {
    console.error('Erro ao gerar relatório de entrada de itens:', error);
    showAlert(error.message || 'Ocorreu um erro ao gerar o relatório de entrada de itens.', 'danger');
  } finally {
    uiService.hideLoading();
  }
}

async _handleGerarReportRetiradasSetor() {
  if (!this.formReportRetiradasSetor.checkValidity()) {
    this.formReportRetiradasSetor.reportValidity();
  }
}

```

```

        return;
    }

    const setor_id = this.selectSetor.value;
    const data_inicio = document.getElementById('dataInicioRetiradaSetor').value;
    const data_fim = document.getElementById('dataFimRetiradaSetor').value;
    const formato = document.getElementById('formatoRetiradasSetor').value;

    if (!setor_id || !data_inicio || !data_fim) {
        showAlert('Por favor, preencha todos os campos obrigatórios.', 'warning');
        return;
    }

    const params = new URLSearchParams({ setor_id, data_inicio, data_fim, formato });

    uiService.showLoading();
    try {
        const token = localStorage.getItem('token');
        const response = await fetch(`/api/almoxarifado/relatorios/retiradas-setor/?${params.toString()}`, {
            method: 'GET',
            headers: {
                'Authorization': `Bearer ${token}`
            }
        });

        if (!response.ok) {
            const errorData = await response.json().catch(() => ({ detail: 'Erro desconhecido ao gerar relatório de retiradas por setor.' }));
            throw new Error(errorData.detail || `Erro HTTP: ${response.status}`);
        }

        const blob = await response.blob();
        const contentDisposition = response.headers.get('content-disposition');
        let filename = `relatorio_retiradas_setor.${formato}`;

        if (contentDisposition) {
            const filenameMatch = contentDisposition.match(/filename\*?=(?:UTF-8'')?([^\s;]+)/i);
            if (filenameMatch && filenameMatch[1]) {
                filename = decodeURIComponent(filenameMatch[1].replace(/"/g, ''));
            } else {
                const simpleSplit = contentDisposition.split('filename=')[1];
                if (simpleSplit) {
                    filename = simpleSplit.replace(/"/g, '');
                }
            }
        }

        this._downloadFile(blob, filename);

        showAlert('Relatório de retiradas por setor gerado com sucesso!', 'success');
        this.modalReportRetiradasSetor.hide();
    } catch (error) {
        console.error('Erro ao gerar relatório de retiradas por setor:', error);
        showAlert(error.message || 'Ocorreu um erro ao gerar o relatório de retiradas por setor.', 'error');
    } finally {
        uiService.hideLoading();
    }
}

async _handleGerarReportRetiradasUsuario() {
    if (!this.formReportRetiradasUsuario.checkValidity()) {
        this.formReportRetiradasUsuario.reportValidity();
        return;
    }
}

```

```

const usuario_id = this.selectUsuario.value;
const data_inicio = document.getElementById('dataInicioRetiradaUsuario').value;
const data_fim = document.getElementById('dataFimRetiradaUsuario').value;
const formato = document.getElementById('formatoRetiradasUsuario').value;

if (!usuario_id || !data_inicio || !data_fim) {
    showAlert('Por favor, preencha todos os campos obrigatórios.', 'warning');
    return;
}

const params = new URLSearchParams({ usuario_id, data_inicio, data_fim, formato });

uiService.showLoading();
try {
    const token = localStorage.getItem('token');
    const response = await fetch(`/api/almoxarifado/relatorios/retiradas-usuario/?${params.toString()}`, {
        method: 'GET',
        headers: {
            'Authorization': `Bearer ${token}`
        }
    });

    if (!response.ok) {
        const errorData = await response.json().catch(() => ({ detail: 'Erro desconhecido ao gerar relatório de retiradas por usuário.' }));
        throw new Error(errorData.detail || `Erro HTTP: ${response.status}`);
    }

    const blob = await response.blob();
    const contentDisposition = response.headers.get('content-disposition');
    let filename = `relatorio_retiradas_usuario.${formato}`;

    if (contentDisposition) {
        const filenameMatch = contentDisposition.match(/filename\*?=(?:UTF-8'')?([^\s;]+)/i);
        if (filenameMatch && filenameMatch[1]) {
            filename = decodeURIComponent(filenameMatch[1].replace(/"/g, ''));
        } else {
            const simpleSplit = contentDisposition.split('filename=')[1];
            if (simpleSplit) {
                filename = simpleSplit.replace(/^"/g, '');
            }
        }
    }

    this._downloadFile(blob, filename);

    showAlert('Relatório de retiradas por usuário gerado com sucesso!', 'success');
    this.modalReportRetiradasUsuario.hide();
} catch (error) {
    console.error('Erro ao gerar relatório de retiradas por usuário:', error);
    showAlert(error.message || 'Ocorreu um erro ao gerar o relatório de retiradas por usuário.', 'error');
} finally {
    uiService.hideLoading();
}

_downloadFile(blob, filename) {
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = filename;
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
}

```

```

        window.URL.revokeObjectURL(url);
    }
}

export const reportsModule = new ReportsModule();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\retiradasModule.js

```

// frontend/static/js/retiradasModule.js

import { apiService } from './apiService.js';
import { dataService } from './dataService.js';
import { uiService } from './uiService.js';
import { showAlert, getStatusText, getStatusValue, getUserTypeFromToken } from './utils.js';
import estadoGlobal from './estadoGlobal.js';

class RetiradasModule {
    constructor() {
        this.historicoPageSizeSelectId = 'historicoPageSize';
        this.pendentesPageSizeSelectId = 'pendentesPageSize';

        // Referências aos elementos do modal de conclusão (NOVO)
        this.modalConcluirRetirada = uiService.getModalInstance('modalConcluirRetirada');
        this.concluirRetiradaIdInput = document.getElementById('concluirRetiradaId');
        this.concluirRetiradaDisplayId = document.getElementById('concluirRetiradaDisplayId');
        this.concluirDetalheStatusInput = document.getElementById('concluirDetalheStatus');
        this.btnConfirmarConcluirRetirada = document.getElementById('btn-confirmar-concluir-retirada');

        // Bindings para eventos que são removidos e adicionados novamente
        this._boundHandleHistoricoPaginationClick = this._handleHistoricoPaginationClick.bind(this);
        this._boundHandleHistoricoPageSizeChange = this._handleHistoricoPageSizeChange.bind(this);
        this._boundHandleHistoricoFilterSubmit = this._handleHistoricoFilterSubmit.bind(this);
        this._boundHandleHistoricoClearFilters = this._handleHistoricoClearFilters.bind(this);
    }

    async renderHistoricoRetiradas(page = 1, filters = estadoGlobal.currentHistoricoFilters, pageSize = uiService.showLoading());
    try {
        const data = await dataService.getProcessedRetiradas(apiService.fetchAllRetiradas.bind(apiService), estadoGlobal.setHistoricoPagination(data.current_page, data.total_pages, pageSize, filters); estadoGlobal.setAllRetiradas(data.items);

        const userType = await getUserTypeFromToken();

        const filterFormHtml = this._getHistoricoFilterFormHtml(filters);

        const tableHeaders = ['ID', 'Usuário', 'Data', 'Status', 'Ações'];

        const tableHtml = uiService.renderTable(tableHeaders, estadoGlobal.allRetiradas, {
            noRecordsMessage: "Nenhum histórico de retirada encontrado.",
            rowMapper: (r) => [
                r.retirada_id,
                r.usuario_nome,
                new Date(r.data_solicitacao).toLocaleDateString('pt-BR'),
                getStatusText(r.status)
            ],
            actionsHtml: (r) => {
                // Container flex para empilhar em telas pequenas e garantir btn-acoes
                let actions = `

```

```

        <div class="d-flex flex-wrap justify-content-center gap-1">
            <button
                class="btn btn-sm btn-primary btn-acoes btn-detalhes-retirada"
                data-id="{r.retirada_id}"
            >
                <i class="bi bi-eye"></i> Detalhes
            </button>`;
        if (r.status === estadoGlobal.statusMapUpdate.AUTORIZADA && userType === 2 ) {
            actions += `
                <button
                    class="btn btn-sm btn-success btn-acoes btn-concluir-retirada-trigger"
                    data-id="{r.retirada_id}"
                >
                    <i class="bi bi-check-circle"></i> Concluir Retirada
                </button>`;
        }
        actions += `
    </div>`;
    return actions;
    }
    });

    const paginationHtml = uiService.renderPagination(
        estadoGlobal.currentHistoricoPage,
        estadoGlobal.totalHistoricoPages,
        'historico',
        this.historicoPageSizeSelectId,
        estadoGlobal.currentHistoricoPageSize
    );

    uiService.renderPage('Histórico de Retiradas', `
        <div class="card mb-4">
            <div class="card-header">Filtros de Busca</div>
            <div class="card-body">${filterFormHtml}</div>
        </div>
        ${tableHtml}
        ${paginationHtml}
    `);

    this._bindHistoricoEvents();
    this._bindCommonRetiradaActions();
    this._bindConcluirRetiradaEvents();
} catch (error) {
    console.error("Erro ao renderizar histórico de retiradas:", error);
    showAlert(error.message || 'Ocorreu um erro ao carregar o histórico de retiradas.', 'danger')
} finally {
    uiService.hideLoading();
}
}

async renderPendentesRetiradas(page = 1, pageSize = estadoGlobal.currentPendentesPageSize) {
    uiService.showLoading();
    try {
        const data = await dataService.getProcessedRetiradas(apiService.fetchRetiradasPendentes.bind(
            estadoGlobal.setPendentesPagination(data.current_page, data.total_pages, pageSize);
            estadoGlobal.setPendentesRetiradas(data.items);

            const tableHeaders = ['ID', 'Usuário', 'Setor', 'Data', 'Ações'];

            const tableHtml = uiService.renderTable(tableHeaders, estadoGlobal.pendentesRetiradas, {
                noRecordsMessage: "Nenhuma retirada pendente encontrada.",
                rowMapper: (r) => [
                    r.retirada_id,

```

```

        r.usuario_nome,
        r.setor_nome,
        new Date(r.data_solicitacao).toLocaleDateString('pt-BR')
    ],
    actionsHtml: (r) => `
        <div class="d-flex flex-wrap justify-content-center gap-1">
            <button
                class="btn btn-sm btn-success btn-acoes btn-autorizar-retirada-trigger"
                data-id="${r.retirada_id}"
            >
                <i class="bi bi-check-circle"></i> Autorizar/Negar
            </button>
            <button
                class="btn btn-sm btn-info btn-acoes btn-detalhes-retirada"
                data-id="${r.retirada_id}"
            >
                <i class="bi bi-eye"></i> Ver detalhes
            </button>
        </div>
    `;

    const paginationHtml = uiService.renderPagination(
        estadoGlobal.currentPendentesPage,
        estadoGlobal.totalPendentesPages,
        'pendentes',
        this.pendentesPageSizeSelectId,
        estadoGlobal.currentPendentesPageSize
    );

    uiService.renderPage('Retiradas Pendentes', `
        ${tableHtml}
        ${paginationHtml}
    `);

    this._bindPendentesEvents();
    this._bindCommonRetiradaActions();
} catch (error) {
    console.error("Erro ao renderizar retiradas pendentes:", error);
    showAlert(error.message || 'Ocorreu um erro ao carregar as retiradas pendentes.', 'danger');
} finally {
    uiService.hideLoading();
}
}

// NOVO: Lógica para o modal de Concluir Retirada
_bindConcluirRetiradaEvents() {
    document.querySelectorAll('.btn-concluir-retirada-trigger').forEach(btn => {
        btn.onclick = () => {
            const id = parseInt(btn.dataset.id);
            const retirada = estadoGlobal.allRetiradas.find(r => r.retirada_id === id);
            if (retirada) {
                uiService.fillModalConcluir(retirada);
                this.modalConcluirRetirada.show();
            } else {
                showAlert('Retirada não encontrada para conclusão.', 'warning');
            }
        };
    });
}

if (this.btnConfirmarConcluirRetirada) {
    this.btnConfirmarConcluirRetirada.onclick = () => {
        const retiradaId = parseInt(this.concluirRetiradaIdInput.value);

```



```

        const detalhe = this.concluirDetalheStatusInput.value.trim();
        this._handleConcluirRetirada(retiradaId, detalhe);
    };
    }
}

_getHistoricoFilterFormHtml(currentFilters) {
    return `
        <form id="form-filter-historico" class="row g-3 mb-0">
            <div class="col-12 col-md">
                <label for="filterStatus" class="form-label">Status</label>
                <select class="form-select" id="filterStatus">
                    <option value="">Todos</option>
                    <option value="PENDENTE" ${currentFilters.status === estadoGlobal.statusMapUpdate.PENDENTE}>PENDENTE</option>
                    <option value="AUTORIZADA" ${currentFilters.status === estadoGlobal.statusMapUpdate.AUTORIZADA}>AUTORIZADA</option>
                    <option value="CONCLUIDA" ${currentFilters.status === estadoGlobal.statusMapUpdate.CONCLUIDA}>CONCLUIDA</option>
                    <option value="NEGADA" ${currentFilters.status === estadoGlobal.statusMapUpdate.NEGADA}>NEGADA</option>
                </select>
            </div>
            <div class="col-12 col-md">
                <label for="filterSolicitante" class="form-label">Solicitante</label>
                <input type="text" class="form-control" id="filterSolicitante" value="${currentFilters.solicitante}" />
            </div>
            <div class="col-12 col-md">
                <label for="filterStartDate" class="form-label">Data Inicial</label>
                <input type="date" class="form-control" id="filterStartDate" value="${currentFilters.startDate}" />
            </div>
            <div class="col-12 col-md">
                <label for="filterEndDate" class="form-label">Data Final</label>
                <input type="date" class="form-control" id="filterEndDate" value="${currentFilters.endDate}" />
            </div>
            <div class="col-12 col-md d-flex justify-content-end align-items-end">
                <button type="submit" class="btn btn-primary me-2" id="btn-search-historico">Buscar</button>
                <button type="button" class="btn btn-secondary" id="btn-clear-filters">Limpar Filtros</button>
            </div>
        </form>
    `;
}

_bindHistoricoEvents() {
    const historicoPaginationNav = document.getElementById('historico-pagination-nav');
    const formFilter = document.getElementById('form-filter-historico');
    const pageSizeSelect = document.getElementById(this.historicoPageSizeSelectId);
    const btnClearFilters = document.getElementById('btn-clear-filters');

    if (historicoPaginationNav) {
        historicoPaginationNav.removeEventListener('click', this._boundHandleHistoricoPaginationClick);
    }
    if (pageSizeSelect) {
        pageSizeSelect.removeEventListener('change', this._boundHandleHistoricoPageSizeChange);
    }
    if (formFilter) {
        formFilter.removeEventListener('submit', this._boundHandleHistoricoFilterSubmit);
    }
    if (btnClearFilters) {
        btnClearFilters.removeEventListener('click', this._boundHandleHistoricoClearFilters);
    }

    if (historicoPaginationNav) {
        historicoPaginationNav.addEventListener('click', this._boundHandleHistoricoPaginationClick);
    }
    if (pageSizeSelect) {
        pageSizeSelect.addEventListener('change', this._boundHandleHistoricoPageSizeChange);
    }
}

```

```

    }
    if (formFilter) {
        formFilter.addEventListener('submit', this._boundHandleHistoricoFilterSubmit);
    }
    if (btnClearFilters) {
        btnClearFilters.addEventListener('click', this._boundHandleHistoricoClearFilters);
    }
}

async _handleHistoricoFilterSubmit(e) {
    e.preventDefault();
    const selectedStatusString = document.getElementById('filterStatus').value;
    const statusInt = selectedStatusString ? estadoGlobal.statusMapUpdate[selectedStatusString] : null;
    const filters = {
        status: statusInt,
        solicitante: document.getElementById('filterSolicitante').value.trim(),
        start_date: document.getElementById('filterStartDate').value,
        end_date: document.getElementById('filterEndDate').value,
    };
    this.renderHistoricoRetiradas(1, filters, estadoGlobal.currentHistoricoPageSize);
}

_handleHistoricoClearFilters() {
    document.getElementById('filterStatus').value = '';
    document.getElementById('filterSolicitante').value = '';
    document.getElementById('filterStartDate').value = '';
    document.getElementById('filterEndDate').value = '';
    this.renderHistoricoRetiradas(1, {}, estadoGlobal.currentHistoricoPageSize);
}

_handleHistoricoPaginationClick(e) {
    e.preventDefault();
    const clickedPageLink = e.target.closest('a[data-page^="historico-"]');
    const clickedActionButton = e.target.closest('a[data-action^="historico-"]');

    if (clickedPageLink) {
        const pageValue = clickedPageLink.dataset.page.split('-')[1];
        const newPage = parseInt(pageValue);
        if (!isNaN(newPage) && newPage !== estadoGlobal.currentHistoricoPage) {
            this.renderHistoricoRetiradas(newPage, estadoGlobal.currentHistoricoFilters, estadoGlobal.currentHistoricoPageSize);
        }
        return;
    }

    if (clickedActionButton) {
        const action = clickedActionButton.dataset.action;
        let newPage = estadoGlobal.currentHistoricoPage;

        if (action === 'historico-prev' && newPage > 1) {
            newPage--;
        } else if (action === 'historico-next' && newPage < estadoGlobal.totalHistoricoPages) {
            newPage++;
        }

        if (newPage !== estadoGlobal.currentHistoricoPage) {
            this.renderHistoricoRetiradas(newPage, estadoGlobal.currentHistoricoFilters, estadoGlobal.currentHistoricoPageSize);
        }
        return;
    }
}

_handleHistoricoPageSizeChange(e) {
    if (e.target.id === this.historicoPageSizeSelectId) {

```

```

        const newPageSize = parseInt(e.target.value);
        this.renderHistoricoRetiradas(1, estadoGlobal.currentHistoricoFilters, newPageSize);
    }
}

_bindPendentesEvents() {
    const pendentesPaginationNav = document.getElementById('pendentes-pagination-nav');
    const pageSizeSelect = document.getElementById(this.pendentesPageSizeSelectId);

    if (!pendentesPaginationNav) return;

    pendentesPaginationNav.removeEventListener('click', this._boundHandlePendentesPaginationClick);
    if (pageSizeSelect) {
        pageSizeSelect.removeEventListener('change', this._boundHandlePendentesPageSizeChange);
    }

    pendentesPaginationNav.addEventListener('click', this._boundHandlePendentesPaginationClick);
    if (pageSizeSelect) {
        pageSizeSelect.addEventListener('change', this._boundHandlePendentesPageSizeChange);
    }
}

_boundHandlePendentesPaginationClick = (e) => {
    e.preventDefault();
    const clickedPageLink = e.target.closest('a[data-page^="pendentes-"]');
    const clickedActionButton = e.target.closest('a[data-action^="pendentes-"]');

    if (clickedPageLink) {
        const pageValue = clickedPageLink.dataset.page.split('-')[1];
        const newPage = parseInt(pageValue);
        if (!isNaN(newPage) && newPage !== estadoGlobal.currentPendentesPage) {
            this.renderPendentesRetiradas(newPage, estadoGlobal.currentPendentesPageSize);
        }
        return;
    }

    if (clickedActionButton) {
        const action = clickedActionButton.dataset.action;
        let newPage = estadoGlobal.currentPendentesPage;

        if (action === 'pendentes-prev' && newPage > 1) {
            newPage--;
        } else if (action === 'pendentes-next' && newPage < estadoGlobal.totalPendentesPages) {
            newPage++;
        }

        if (newPage !== estadoGlobal.currentPendentesPage) {
            this.renderPendentesRetiradas(newPage, estadoGlobal.currentPendentesPageSize);
        }
        return;
    }
}

_boundHandlePendentesPageSizeChange = (e) => {
    if (e.target.id === this.pendentesPageSizeSelectId) {
        const newPageSize = parseInt(e.target.value);
        this.renderPendentesRetiradas(1, newPageSize);
    }
}

_bindCommonRetiradaActions() {
    document.querySelectorAll('.btn-detalhes-retirada').forEach(btn => {
        btn.removeEventListener('click', this._boundHandleDetalhesRetiradaClick);
    });
}

```

```

    });
    document.querySelectorAll('.btn-autorizar-retirada-trigger').forEach(btn => {
        btn.removeEventListener('click', this._boundHandleAutorizarRetiradaClick);
    });

    document.querySelectorAll('.btn-detahes-retirada').forEach(btn => {
        btn.addEventListener('click', this._boundHandleDetahesRetiradaClick);
    });

    document.querySelectorAll('.btn-autorizar-retirada-trigger').forEach(btn => {
        btn.addEventListener('click', this._boundHandleAutorizarRetiradaClick);
    });
}

_boundHandleDetahesRetiradaClick = (e) => {
    const id = parseInt(e.currentTarget.dataset.id);
    const retirada = [...estadoGlobal.allRetiradas, ...estadoGlobal.pendentesRetiradas].find(r => r.id === id);
    if (retirada) {
        uiService.fillModalDetahes(retirada);
        uiService.getModalInstance('modalVerDetahesRetirada').show();
    } else {
        showAlert('Detahes da retirada não encontrados.', 'warning');
    }
}

_boundHandleAutorizarRetiradaClick = (e) => {
    const id = parseInt(e.currentTarget.dataset.id);
    const retirada = estadoGlobal.pendentesRetiradas.find(x => x.retirada_id === id);
    if (retirada) {
        uiService.fillModalAutorizar(retirada);
        uiService.getModalInstance('modalAutorizarRetirada').show();
    } else {
        showAlert('Retirada pendente não encontrada para autorização.', 'warning');
    }
}

async _handleAuthorizeDeny(action, e) {
    const id = parseInt(e.currentTarget.dataset.id);
    const detalheInput = document.getElementById('autorizarDetalheStatus');
    const detalhe = detalheInput.value.trim();
    const statusValue = getStatusValue(action);

    if (action === 'NEGADA' && !detalhe) {
        showAlert('O detalhe do status (justificativa da negação) é obrigatório ao negar.', 'warning');
        detalheInput.focus();
        return;
    }

    try {
        await apiService.updateRetiradaStatus(id, statusValue, detalhe);
        showAlert(`Retirada ${action === 'AUTORIZADA' ? 'autorizada' : 'negada'} com sucesso!`, 'success');
        uiService.getModalInstance('modalAutorizarRetirada').hide();
        this.renderPendentesRetiradas(estadoGlobal.currentPendentesPage, estadoGlobal.currentPendentesPage);
    } catch (error) {
        showAlert(error.message || `Erro ao ${action === 'AUTORIZADA' ? 'autorizar' : 'negar'} retirada`, 'error');
    }
}

async _handleConcluirRetirada(retiradaId, detalhe) {
    uiService.showLoading();
    try {
        await apiService.updateRetiradaStatus(retiradaId, estadoGlobal.statusMapUpdate.CONCLUIDA, detalhe);
        showAlert('Retirada marcada como "Concluída" com sucesso! Estoque decrementado.', 'success');
    } catch (error) {
        showAlert(error.message || 'Erro ao concluir retirada', 'error');
    }
}

```

```

        this.modalConcluirRetirada.hide();
        this.renderHistoricoRetiradas(estadoGlobal.currentHistoricoPage, estadoGlobal.currentHistoricoPage);
    } catch (error) {
        console.error("Erro ao concluir retirada", error);
        showAlert(error.message || 'Erro ao concluir a retirada. Verifique o estoque!', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

export const retiradasModule = new RetiradasModule();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\selecionar-item-module.js

```

// frontend/static/js/selecionar-item-module.js

import { apiService } from '../apiService.js';
import { uiService } from '../uiService.js';
import { showAlert } from '../utils.js';

class SelecionarItemModule {
    constructor() {
        this.modalSelecionarItem = uiService.getModalInstance('modalSelecionarItemRetirada');
        this.searchForm = document.getElementById('form-search-item-retirada');
        this.searchItemNameInput = document.getElementById('searchItemName');
        this.btnSearchItem = document.getElementById('btn-search-item');
        this.btnClearItemSearch = document.getElementById('btn-clear-item-search');
        this.itensListContainer = document.getElementById('itens-list-container');
        this.itensPaginationContainer = document.getElementById('itens-pagination-container');

        this.currentPage = 1;
        this.pageSize = 10; // Default page size
        this.currentSearchTerm = '';
        this.callerEventName = 'itemSelectedForRetirada';
        this.currentTotalPages = 1; // Novo atributo para armazenar o total de páginas
    }

    init() {
        this.bindEvents();
    }

    bindEvents() {
        this.searchForm.addEventListener('submit', (e) => {
            e.preventDefault();
            this.currentSearchTerm = this.searchItemNameInput.value.trim();
            this.currentPage = 1;
            this.loadItems();
        });

        this.btnClearItemSearch.addEventListener('click', () => {
            this.currentSearchTerm = '';
            this.searchItemNameInput.value = '';
            this.currentPage = 1;
            this.loadItems();
        });

        this.itensPaginationContainer.addEventListener('click', (e) => {
            e.preventDefault(); // Previne o comportamento padrão do link para todos os cliques de paginação
            // Lógica para links de página individuais (números)

```

```

const clickedPageLink = e.target.closest('a[data-page]');
if (clickedPageLink) {
    // Extrair o número da página corretamente da string "itemSearch-N"
    const pageValue = clickedPageLink.dataset.page.split('-')[1]; // Pega "N" de "itemSearch-N"
    const parsedPage = parseInt(pageValue);

    if (isNaN(parsedPage)) {
        console.error('ERRO: parseInt retornou NaN para o valor de pageValue:', pageValue);
        showAlert('Erro na paginação: o número da página não é válido.', 'danger');
        return;
    }

    this.currentPage = parsedPage;
    this.loadItems();
    return;
}

// Lógica para botões "Anterior" ou "Próximo"
const clickedActionButton = e.target.closest('a[data-action]');
if (clickedActionButton) {
    const action = clickedActionButton.dataset.action;
    let newPage = this.currentPage;
    const totalPages = this.currentTotalPages;

    if (action.includes('prev')) {
        if (newPage > 1) newPage--;
    } else if (action.includes('next')) {
        if (newPage < totalPages) newPage++;
    }

    if (newPage !== this.currentPage) {
        this.currentPage = newPage;
        this.loadItems();
    }
}
});

this.itensPaginationContainer.addEventListener('change', (e) => {
    if (e.target.id === 'itemSearchPageSize') {
        this.pageSize = parseInt(e.target.value);
        this.currentPage = 1; // Reseta para a primeira página
        this.loadItems();
    }
});

this.itensListContainer.addEventListener('click', (e) => {
    if (e.target.classList.contains('btn-selecionar-item')) {
        const itemId = parseInt(e.target.dataset.id);
        this.selectItem(itemId);
    }
});

document.getElementById('modalSelecionarItemRetirada').addEventListener('hidden.bs.modal', () => {
    this.searchItemNameInput.value = '';
    this.currentSearchTerm = '';
    this.currentPage = 1;
    this.itensListContainer.innerHTML = '';
    this.itensPaginationContainer.innerHTML = '';
    this.callerEventName = 'itemSelectedForRetirada'; // Reset to default
    this.currentTotalPages = 1; // Reset total pages on modal hide
});
}

async openModal(eventName = 'itemSelectedForRetirada', pageSize = this.pageSize) {

```

```

    this.callerEventName = eventName;
    this.pageSize = pageSize;
    this.currentPage = 1;
    this.currentSearchTerm = '';
    this.searchItemNameInput.value = '';

    await this.loadItems();
    this.modalSelecionarItem.show();
  }

  async loadItems() {
    uiService.showLoading();
    try {
      const data = await apiService.searchItems(this.currentSearchTerm, null, this.currentPage, this.pageSize);
      this.currentTotalPages = data.total_pages;

      const tableHeaders = ['Nome', 'Ações'];
      this.itensListContainer.innerHTML = uiService.renderTable(tableHeaders, data.items, {
        noRecordsMessage: "Nenhum item encontrado.",
        rowMapper: (item) => [item.nome_item_original],
        actionsHtml: (item) => `
          <button class="btn btn-sm btn-primary btn-selecionar-item" data-id="${item.item_id}">
            Selecionar
          </button>
        `
      });

      this.itensPaginationContainer.innerHTML = uiService.renderPagination(
        data.page,
        data.total_pages,
        'itemSearch', // Este 'type' ainda será usado para data-action="itemSearch-prev/next"
        'itemSearchPageSize',
        this.pageSize
      );
    } catch (error) {
      console.error("Erro ao carregar itens para seleção", error);
      showAlert(error.message || 'Erro ao carregar itens disponíveis.', 'danger');
      this.itensListContainer.innerHTML = `<p class="text-danger">Não foi possível carregar os itens.</p>`;
      this.itensPaginationContainer.innerHTML = '';
    } finally {
      uiService.hideLoading();
    }
  }

  async selectItem(itemId) {
    uiService.showLoading();
    try {
      const item = await apiService.getItemById(itemId);

      const event = new CustomEvent(this.callerEventName, {
        detail: { item: item }
      });
      document.dispatchEvent(event);

      this.modalSelecionarItem.hide();
      showAlert(`Item "${item.nome_item_original}" selecionado com sucesso!`, 'success');
    } catch (error) {
      console.error("Erro ao selecionar item:", error);
      showAlert(error.message || 'Erro ao selecionar o item.', 'danger');
    } finally {
      uiService.hideLoading();
    }
  }

```

```

    }
  }
}

export const selecionarItemModule = new SelecionarItemModule();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\setoresModule.js

```

// frontend/static/js/setoresModule.js

import { apiService } from './apiService.js';
import { uiService } from './uiService.js';
import { showAlert } from './utils.js';

class SetoresModule {
  constructor() {
    this.currentPage = 1;
    this.pageSize = 10; // Default page size
    this.searchNome = ''; // Current search term for sector name
    this.totalSetores = 0; // To store total sectors for pagination calculation

    // Modals
    this.modalCadastrarSetor = uiService.getModalInstance('modalCadastrarSetor');
    this.modalEditarSetor = uiService.getModalInstance('modalEditarSetor');
    this.modalConfirmarDeleteSetor = uiService.getModalInstance('modalConfirmarDeleteSetor');

    // Forms
    this.formCadastrarSetor = document.getElementById('form-cadastrar-setor');
    this.formEditarSetor = document.getElementById('form-editar-setor');

    // Buttons
    this.btnSalvarCadastrarSetor = document.getElementById('btn-salvar-cadastrar-setor');
    this.btnSalvarEditarSetor = document.getElementById('btn-salvar-editar-setor');
    this.btnConfirmarDeletarSetor = document.getElementById('btn-confirmar-deletar-setor');

    // Bindings for event listeners to avoid duplication
    this.boundHandlePaginationClick = this._handlePaginationClick.bind(this);
    this.boundHandlePageSizeChange = this._handlePageSizeChange.bind(this);
    this.boundHandleSearchSetores = this._handleSearchSetores.bind(this);
    this.boundHandleClearSearch = this._handleClearSearch.bind(this);
    this.boundHandleTableActions = this._handleTableActions.bind(this);
    this.boundUpdateSetor = this._updateSetor.bind(this); // Bind update method
  }

  init() {
    this.bindEvents();
  }

  bindEvents() {
    // Cadastrar Setor
    document.getElementById('btn-open-cadastrar-setor')?.addEventListener('click', e => {
      e.preventDefault();
      this.formCadastrarSetor.reset();
      this.modalCadastrarSetor.show();
    });

    this.btnSalvarCadastrarSetor?.addEventListener('click', () => this._createSetor());

    // Confirmar Deleção
    this.btnConfirmarDeletarSetor?.addEventListener('click', this._deleteSetorConfirmed.bind(this));
    // Salvar Edição

```



```

    this.btnSalvarEditarSetor?.addEventListener('click', this.boundUpdateSetor);
  }

  async renderSetoresList() {
    uiService.showLoading();
    try {
      // Fetch all sectors. For simplicity, fetching all and then filtering/paginating in frontend
      // For large datasets, backend pagination/filtering would be more efficient.
      const allSetores = await apiService.get('/setores');

      const filteredSetores = this.searchNome
        ? allSetores.filter(s => s.nome_setor.toLowerCase().includes(this.searchNome.toLowerCase()))
        : allSetores;

      this.totalSetores = filteredSetores.length; // Update total sectors count
      const totalPages = Math.ceil(this.totalSetores / this.pageSize);

      const offset = (this.currentPage - 1) * this.pageSize;
      const setoresForPage = filteredSetores.slice(offset, offset + this.pageSize);

      const tableHeaders = ['ID', 'Nome', 'Descrição', 'Ações'];
      const tableHtml = uiService.renderTable(tableHeaders, setoresForPage, {
        noRecordsMessage: 'Nenhum setor encontrado.',
        rowMapper: (setor) => [
          setor.setor_id,
          setor.nome_setor,
          setor.descricao_setor || '-'
        ],
        actionsHtml: (setor) => `
          <div class="d-flex flex-wrap justify-content-center gap-1">
            <button class="btn btn-sm btn-primary btn-acoas btn-editar-setor" data-id="${setor.setor_id}">
              <i class="bi bi-pencil-square"></i> Editar
            </button>
            <button class="btn btn-sm btn-danger btn-acoas btn-deletar-setor" data-id="${setor.setor_id}">
              <i class="bi bi-trash"></i> Deletar
            </button>
          </div>
        `
      });

      const paginationHtml = uiService.renderPagination(
        this.currentPage,
        totalPages,
        'setores', // type for pagination links
        'setoresPageSizeSelect', // id for pageSize select element
        this.pageSize
      );

      const searchBarHtml = `
        <div class="card mb-3">
          <div class="card-header">Filtros de Busca</div>
          <div class="card-body">
            <form id="search-bar-setores" class="row g-3 mb-0">
              <div class="col-12 col-md">
                <label for="search-setor-nome" class="form-label">Nome do Setor</label>
                <input type="text" id="search-setor-nome" class="form-control" placeholder="Digite o nome do setor">
              </div>
              <div class="col-12 col-md d-flex justify-content-end align-items-end">
                <button type="button" id="btn-search-setor" class="btn btn-primary me-2">Pesquisar</button>
                <button type="button" id="btn-clear-search-setor" class="btn btn-secondary">Limpar</button>
              </div>
            </form>
          </div>
        </div>
      `;
    } catch (error) {
      console.error('Erro ao buscar setores:', error);
    }
  }

```

```

        </div>
    `;

    uiService.renderPage('Gerenciamento de Setores', `
        ${searchBarHtml}
        ${tableHtml}
        ${paginationHtml}
    `);

    this._bindPageEvents(); // Re-bind events after content is rendered
    this._bindTableActions(); // Re-bind table action events

} catch (error) {
    console.error('Erro ao renderizar lista de setores', error);
    showAlert(error.message || 'Erro ao carregar setores.', 'danger');
    uiService.renderPage('Gerenciamento de Setores', `<div class="alert alert-warning">Erro ao c
} finally {
    uiService.hideLoading();
}
}

_bindPageEvents() {
    const paginationNav = document.getElementById('setores-pagination-nav');
    const pageSizeSelect = document.getElementById('setoresPageSizeSelect');
    const btnClearSearch = document.getElementById('btn-clear-search-setor');
    const btnSearch = document.getElementById('btn-search-setor');

    // Remove previous listeners to prevent duplication
    if (paginationNav) {
        paginationNav.removeEventListener('click', this.boundHandlePaginationClick);
    }
    if (pageSizeSelect) {
        pageSizeSelect.removeEventListener('change', this.boundHandlePageSizeChange);
    }
    if (btnSearch) {
        btnSearch.removeEventListener('click', this.boundHandleSearchSetores);
    }
    if (btnClearSearch) {
        btnClearSearch.removeEventListener('click', this.boundHandleClearSearch);
    }

    // Add new listeners
    if (paginationNav) {
        paginationNav.addEventListener('click', this.boundHandlePaginationClick);
    }
    if (pageSizeSelect) {
        pageSizeSelect.addEventListener('change', this.boundHandlePageSizeChange);
    }
    if (btnSearch) {
        btnSearch.addEventListener('click', this.boundHandleSearchSetores);
    }
    if (btnClearSearch) {
        btnClearSearch.addEventListener('click', this.boundHandleClearSearch);
    }
}

_handlePaginationClick(e) {
    e.preventDefault();

    const clickedPageLink = e.target.closest('a[data-page^="setores-"]');
    const clickedActionButton = e.target.closest('a[data-action^="setores-"]');

    if (clickedPageLink) {

```

```

        const pageValue = clickedPageLink.dataset.page.split('-')[1];
        const newPage = parseInt(pageValue);
        const totalPages = Math.ceil(this.totalSetores / this.pageSize); // Recalculate based on current page size

        if (!isNaN(newPage) && newPage !== this.currentPage && newPage >= 1 && newPage <= totalPages) {
            this.currentPage = newPage;
            this.renderSetoresList();
        }
        return;
    }

    if (clickedActionButton) {
        const action = clickedActionButton.dataset.action;
        let newPage = this.currentPage;
        const totalPages = Math.ceil(this.totalSetores / this.pageSize); // Recalculate based on current page size

        if (action === 'setores-prev' && newPage > 1) {
            newPage--;
        } else if (action === 'setores-next' && newPage < totalPages) {
            newPage++;
        }

        if (newPage !== this.currentPage) {
            this.currentPage = newPage;
            this.renderSetoresList();
        }
        return;
    }
}

_handlePageSizeChange(e) {
    this.pageSize = parseInt(e.target.value);
    this.currentPage = 1; // Reset to first page when page size changes
    this.renderSetoresList();
}

_handleSearchSetores(e) {
    e.preventDefault();
    this.searchNome = document.getElementById('search-setor-nome').value.trim();
    this.currentPage = 1; // Reset to first page on new search
    this.renderSetoresList();
}

_handleClearSearch(e) {
    e.preventDefault();
    document.getElementById('search-setor-nome').value = '';
    this.searchNome = '';
    this.currentPage = 1; // Reset to first page on clear search
    this.renderSetoresList();
}

_bindTableActions() {
    const mainContent = document.getElementById('main-content');
    if (mainContent) {
        mainContent.removeEventListener('click', this.boundHandleTableActions); // Remove old listener
        mainContent.addEventListener('click', this.boundHandleTableActions); // Add new listener
    }
}

async _handleTableActions(e) {
    const editButton = e.target.closest('.btn-editar-setor');
    const deleteButton = e.target.closest('.btn-deletar-setor');
    if (editButton) {

```

```

        const setId = parseInt(editButton.dataset.id);
        await this._openEditModal(setId);
    } else if (deleteButton) {
        const setId = parseInt(deleteButton.dataset.id);
        this._openDeleteConfirmModal(setId);
    }
}

async _createSetor() {
    if (!this.formCadastrarSetor.checkValidity()) {
        this.formCadastrarSetor.reportValidity();
        return;
    }

    const formData = new FormData(this.formCadastrarSetor);
    const setIdData = {};
    for (const [key, value] of formData.entries()) {
        setIdData[key] = value;
    }

    uiService.showLoading();
    try {
        await apiService.post('/setores/', setIdData);
        showAlert('Setor cadastrado com sucesso!', 'success');
        this.modalCadastrarSetor.hide();
        this.renderSetoresList(); // Refresh list
    } catch (error) {
        console.error('Erro ao cadastrar setor', error);
        showAlert(error.message || 'Erro ao cadastrar setor.', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

async _openEditModal(setId) {
    uiService.showLoading();
    try {
        const setor = await apiService.get(`/setores/${setId}`);
        this.formEditarSetor.querySelector('input[name="nome_setor"]').value = setor.nome_setor;
        this.formEditarSetor.querySelector('textarea[name="descricao_setor"]').value = setor.descricao_setor;

        // Store setor ID in the save button for later use
        this.btnSalvarEditarSetor.dataset.id = setId;
        this.modalEditarSetor.show();
    } catch (error) {
        console.error('Erro ao carregar dados do setor para edição', error);
        showAlert(error.message || 'Erro ao carregar dados do setor.', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

async _updateSetor() {
    const setId = parseInt(this.btnSalvarEditarSetor.dataset.id);
    if (!setId) {
        showAlert('ID do setor para edição não encontrado.', 'danger');
        return;
    }

    if (!this.formEditarSetor.checkValidity()) {
        this.formEditarSetor.reportValidity();
        return;
    }
}

```

```

const formData = new FormData(this.formEditarSetor);
const setData = {};
for (const [key, value] of formData.entries()) {
    setData[key] = value;
}

uiService.showLoading();
try {
    await apiService.put(`/setores/${setorId}`, setData); // Use PUT for full update
    showAlert('Setor atualizado com sucesso!', 'success');
    this.modalEditarSetor.hide();
    this.renderSetoresList(); // Refresh list
} catch (error) {
    console.error('Erro ao atualizar setor', error);
    showAlert(error.message || 'Erro ao atualizar setor.', 'danger!');
} finally {
    uiService.hideLoading();
}
}

_openDeleteConfirmModal(setorId) {
    document.getElementById('confirm-delete-setor-id').textContent = setorId;
    this.btnConfirmarDeletarSetor.dataset.id = setorId; // Store ID for confirmation
    this.modalConfirmarDeleteSetor.show();
}

async _deleteSetorConfirmed() {
    const setorId = parseInt(this.btnConfirmarDeletarSetor.dataset.id);
    if (!setorId) {
        showAlert('ID do setor para exclusão não encontrado.', 'danger!');
        return;
    }

    uiService.showLoading();
    try {
        await apiService.delete(`/setores/${setorId}`);
        showAlert('Setor deletado com sucesso!', 'success');
        this.modalConfirmarDeleteSetor.hide();
        this.renderSetoresList(); // Refresh list
    } catch (error) {
        console.error('Erro ao deletar setor', error);
        showAlert(error.message || 'Erro ao deletar setor.', 'danger!');
    } finally {
        uiService.hideLoading();
    }
}

}

export const setoresModule = new SetoresModule();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\solicitar-retirada.js

```

// frontend/static/js/solicitar-retirada.js

import { apiService } from '../apiService.js';
import { uiService } from '../uiService.js';
import { showAlert, getUserIdFromToken } from '../utils.js'; // Import getUserIdFromToken
import { selecionarItemModule } from '../selecionar-item-module.js';

class SolicitarRetiradaModule {

```

```

constructor() {
  this.modalSolicitarRetirada = uiService.getModalInstance('modalSolicitarRetirada');
  this.form = document.getElementById('form-solicitar-retirada');
  this.selectSetor = document.getElementById('solicitar_setor_id');
  this.inputSolicitadoLocalmentePor = document.getElementById('solicitado_localmente_por');
  this.inputJustificativa = document.getElementById('solicitar_justificativa');
  this.btnAbrirSelecionarItemModal = document.getElementById('btn-abrir-selecionar-item-modal');
  this.inputQuantidadeAddItem = document.getElementById('quantidade_add_item');
  this.btnAdicionarItemRetirada = document.getElementById('btn-adicionar-item-retirada');
  this.itensParaRetiradaContainer = document.getElementById('itens-para-retirada-container');
  this.btnSalvarSolicitacaoRetirada = document.getElementById('btn-salvar-solicitacao-retirada');
  this.noItemsMessage = document.getElementById('no-items-message');

  // NOVAS REFERÊNCIAS PARA FEEDBACK VISUAL
  this.selectedItemDisplay = document.getElementById('selected-item-display');
  this.selectedItemName = document.getElementById('selected-item-name');

  this.itensSelecionados = [];
  this.todosItensDisponiveis = [];
  this.currentItemToAddToCart = null;
}

init() {
  this.bindEvents();
}

bindEvents() {
  // Link para abrir o modal de solicitação (geral, por exemplo, do Almoxarifado)
  // O evento para o dashboard do servidor é tratado em main.js
  document.getElementById('btn-open-solicitar-retirada')?.addEventListener('click', e => {
    e.preventDefault();
    this.openModal(false); // Explicitamente false para dashboards que não são de servidor
  });

  this.btnAbrirSelecionarItemModal?.addEventListener('click', () => {
    this.openSelecionarItemModal();
  });

  this.btnSalvarSolicitacaoRetirada.addEventListener('click', () => this._enviarSolicitacao());
  this.btnAdicionarItemRetirada?.addEventListener('click', () => this._adicionarItemParaRetirada());

  document.getElementById('modalSolicitarRetirada').addEventListener('hidden.bs.modal', () => {
    this.resetForm(); // Resetar formulário ao fechar o modal
  });

  document.addEventListener('itemSelectedForRetirada', (event) => {
    this.handleItemSelected(event.detail.item);
  });
}

// MODIFICADO: Adicionado parâmetro isServidorDashboard
async openModal(isServidorDashboard = false) {
  uiService.showLoading();
  try {
    // Sempre busca todos os setores. O comportamento de seleção/desativação será condicional.
    const setoresData = await apiService.fetchAllSetores();

    // Limpa as opções anteriores do select
    this.selectSetor.innerHTML = '';

    if (isServidorDashboard) {
      const userId = getUserIdFromToken();
      if (userId) {

```

```

const userDetails = await apiService.getCurrentUserDetails(userId);
const userSectorId = userDetails.sectorId;

if (userSectorId) {
  // Encontra o setor específico do usuário nos dados carregados
  const userSector = setoresData.find(s => s.setor_id === userSectorId);
  if (userSector) {
    const option = document.createElement('option');
    option.value = userSector.setor_id;
    option.textContent = userSector.nome_setor;
    option.selected = true; // Pré-seleciona o setor do usuário
    this.selectSetor.appendChild(option);
    this.selectSetor.disabled = true; // Torna o campo somente leitura
  } else {
    // Se o setor do usuário não for encontrado na lista, permite seleção manual
    this.selectSetor.innerHTML = '<option value="" disabled selected>Erro: Setor não encontrado';
    setoresData.forEach(setor => {
      const option = document.createElement('option');
      option.value = setor.setor_id;
      option.textContent = setor.nome_setor;
      this.selectSetor.appendChild(option);
    });
    this.selectSetor.disabled = false; // Garante que esteja habilitado se o setor for encontrado
    showAlert('Setor do usuário não encontrado na lista de setores. Por favor, selecione um setor válido');
  }
} else {
  // Se o ID do setor do usuário for nulo/indefinido, permite seleção manual e alerta
  this.selectSetor.innerHTML = '<option value="" disabled selected>Erro: Não foi possível obter o ID do setor do usuário logado. Por favor, selecione o setor do usuário logado';
  setoresData.forEach(setor => {
    const option = document.createElement('option');
    option.value = setor.setor_id;
    option.textContent = setor.nome_setor;
    this.selectSetor.appendChild(option);
  });
  this.selectSetor.disabled = false; // Garante que esteja habilitado
  showAlert('Não foi possível obter o ID do usuário logado. Por favor, selecione o setor do usuário logado');
}
} else {
  // Se o ID do usuário não for obtido, permite seleção manual e alerta
  this.selectSetor.innerHTML = '<option value="" disabled selected>Erro: Não foi possível obter o ID do usuário logado. Por favor, faça login';
  setoresData.forEach(setor => {
    const option = document.createElement('option');
    option.value = setor.setor_id;
    option.textContent = setor.nome_setor;
    this.selectSetor.appendChild(option);
  });
  this.selectSetor.disabled = false; // Garante que esteja habilitado
  showAlert('Não foi possível obter o ID do usuário logado. Por favor, selecione o setor do usuário logado');
}
} else {
  // Comportamento padrão para dashboards que não são de servidor (Almoxarifado/Direção)
  this.selectSetor.innerHTML = '<option value="" disabled selected>Selecione um setor</option>';
  setoresData.forEach(setor => {
    const option = document.createElement('option');
    option.value = setor.setor_id;
    option.textContent = setor.nome_setor;
    this.selectSetor.appendChild(option);
  });
  this.selectSetor.disabled = false; // Garante que esteja habilitado para não-servidores
}
}

// Lógica para esconder/mostrar o campo "Solicitado localmente por"
const solicitadoLocalmentePorContainer = this.inputSolicitadoLocalmentePor.closest('.col-md-12');

```

```

        if (solicitadoLocalmentePorContainer) {
            if (isServidorDashboard) {
                solicitadoLocalmentePorContainer.style.display = 'none';
            } else {
                solicitadoLocalmentePorContainer.style.display = 'block'; // Ou 'flex' dependendo do
            }
        }
    }

    this.itensSelecionados = [];
    this._renderItensParaRetirada();
    this._clearSelectedItemDisplay();
    uiService.hideLoading();
    this.modalSolicitarRetirada.show();

} catch (error) {
    uiService.hideLoading();
    console.error("Erro ao carregar dados iniciais para solicitação", error);
    showAlert('Erro ao carregar setores. Tente novamente.', 'danger');
    this.modalSolicitarRetirada.hide();
}

}

openSelecionarItemModal() {
    selecionarItemModule.openModal();
}

handleItemSelected(selectedItem) {
    this.currentItemToAddToCart = selectedItem;
    this.inputQuantidadeAddItem.value = 1;

    // ATUALIZA O FEEDBACK VISUAL
    this.selectedItemName.textContent = selectedItem.nome_item_original;
    this.selectedItemDisplay.style.display = 'block';
    this.btnAdicionarItemRetirada.disabled = false;

    showAlert(`Item selecionado: ${selectedItem.nome_item_original}. Agora, informe a quantidade.`);
}

_adicionarItemParaRetirada() {
    if (!this.currentItemToAddToCart) {
        showAlert('Nenhum item selecionado para adicionar.', 'warning');
        return;
    }

    const quantidade = parseInt(this.inputQuantidadeAddItem.value);

    if (isNaN(quantidade) || quantidade <= 0) {
        showAlert('Por favor, informe uma quantidade válida.', 'warning');
        return;
    }

    const itemExistente = this.itensSelecionados.find(item =>
        item.item_id === this.currentItemToAddToCart.item_id
    );

    const quantidadeDisponivel = this.currentItemToAddToCart.quantidade_item;
    const quantidadeTotalSolicitada = quantidade + (itemExistente ? itemExistente.quantidade_retirada : 0);

    if (quantidadeTotalSolicitada > quantidadeDisponivel) {
        showAlert(
            `Quantidade solicitada para ${this.currentItemToAddToCart.nome_item_original} excede o e
            'warning'
        );
    }
}

```



```

        );
        return;
    }

    if (itemExistente) {
        itemExistente.quantidade_retirada += quantidade;
    } else {
        this.itensSelecionados.push({
            item_id: this.currentItemToAddToCart.item_id,
            nome_item: this.currentItemToAddToCart.nome_item_original,
            quantidade_retirada: quantidade
        });
    }

    this._renderItensParaRetirada();
    this._clearSelectedItemDisplay();
    this.currentItemToAddToCart = null;
    this.inputQuantidadeAddItem.value = 1;
}

_renderItensParaRetirada() {
    this.itensParaRetiradaContainer.innerHTML = '';

    if (this.itensSelecionados.length === 0) {
        this.noItemsMessage.style.display = 'block';
        this.itensParaRetiradaContainer.appendChild(this.noItemsMessage);
        return;
    }

    this.noItemsMessage.style.display = 'none';

    this.itensSelecionados.forEach(item => {
        const itemElement = document.createElement('div');
        itemElement.className = 'col-12 d-flex justify-content-between align-items-center bg-light p-2';
        itemElement.innerHTML = `
            <span>${item.nome_item} (Quantidade: ${item.quantidade_retirada})</span>
            <button type="button" class="btn btn-sm btn-danger btn-remove-item" data-item-id="${item.item_id}">
                <i class="bi bi-trash"></i> Remover
            </button>
        `;
        this.itensParaRetiradaContainer.appendChild(itemElement);
    });

    this.itensParaRetiradaContainer.querySelectorAll('.btn-remove-item').forEach(btn => {
        btn.addEventListener('click', (e) => {
            this._removerItemParaRetirada(parseInt(e.currentTarget.dataset.itemId))
        });
    });
}

_removerItemParaRetirada(itemId) {
    this.itensSelecionados = this.itensSelecionados.filter(item =>
        item.item_id !== itemId
    );
    this._renderItensParaRetirada();
}

async _enviarSolicitacao() {
    // Se o campo de setor estiver desabilitado (para servidor),
    // ele já estará pré-selecionado com o setor do usuário logado.
    const setorId = parseInt(this.selectSetor.value);
    const justificativa = this.inputJustificativa.value.trim();
    // O campo 'solicitado_localmente_por' só será pego se estiver visível/habilitado.

```

```

const solicitadoLocalmentePor = this.inputSolicitadoLocalmentePor.value.trim();

if (isNaN(setorId)) {
  showAlert('Por favor, selecione um setor.', 'warning');
  return;
}

if (this.itensSelecionados.length === 0) {
  showAlert('Por favor, adicione pelo menos um item para solicitar a retirada.', 'warning');
  return;
}

uiService.showLoading();
try {
  const retiradaData = {
    setor_id: setorId,
    justificativa: justificativa || null,
    // Inclui solicitado_localmente_por apenas se o campo estiver visível (e, portanto, preenchido)
    solicitado_localmente_por: solicitadoLocalmentePor || null,
    itens: this.itensSelecionados.map(item => ({
      item_id: item.item_id,
      quantidade_retirada: item.quantidade_retirada
    })))
  };

  await apiService.solicitarRetirada(retiradaData);
  showAlert('Solicitação de retirada enviada com sucesso!', 'success');
  this.modalSolicitarRetirada.hide();
  this._resetForm();

  // Recarregar dashboard após envio bem-sucedido
  if (typeof window.loadDashboardOverview === 'function') {
    window.loadDashboardOverview();
  }

} catch (error) {
  console.error("Erro ao enviar solicitação de retirada", error);
  showAlert(error.message || 'Erro ao enviar a solicitação de retirada.', 'danger');
} finally {
  uiService.hideLoading();
}

_resetForm() {
  this.form.reset();
  this.itensSelecionados = [];
  this._renderItensParaRetirada(); // Chamada para garantir que a mensagem 'Nenhum item adicionado' seja exibida
  this.noItemsMessage.style.display = 'block'; // Garante que a mensagem esteja visível após o reset
  this.inputQuantidadeAddItem.value = 1;
  this.currentItemToAddToCart = null;
  this._clearSelectedItemDisplay();
  this.selectSetor.disabled = false; // Garante que o select não fique desabilitado permanentemente

  // Garante que o campo "Solicitado localmente por" seja reexibido
  const solicitadoLocalmentePorContainer = this.inputSolicitadoLocalmentePor.closest('.col-md-6');
  if (solicitadoLocalmentePorContainer) {
    solicitadoLocalmentePorContainer.style.display = 'block';
  }
}

_clearSelectedItemDisplay() {
  this.selectedItemName.textContent = "";
  this.selectedItemDisplay.style.display = 'none';
}

```

```

        this.btnAdicionarItemRetirada.disabled = true;
    }
}

export const solicitarRetiradaModule = new SolicitarRetiradaModule();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\uiService.js

```

// frontend/static/js/uiService.js

import { formatDate, formatDateTime, getStatusText, showAlert, getUserTypeFromToken } from './utils.js';
import estadoGlobal from './estadoGlobal.js';

class UiService {
    constructor() {
        this.mainContent = document.getElementById('main-content');
        this.loadingSpinner = document.getElementById('loading-spinner');
    }

    renderPage(title, contentHtml) {
        if (this.mainContent) {
            this.mainContent.innerHTML = `<h3 class="mb-3">${title}</h3>${contentHtml}`;
        } else {
            console.error("Elemento 'main-content' não encontrado.");
        }
    }

    renderTable(headers, rows, options = {}) {
        const { tableId, noRecordsMessage = 'Nenhum registro encontrado.', rowMapper, actionsHtml = () => '' } = options;

        const tableContent =
            rows.length > 0 ? rows.map(item => `
                <tr>
                    ${rowMapper(item).map(cell => `<td>${cell}</td>`).join('')}
                    <td>${actionsHtml(item)}</td>
                </tr>
            `).join('') : `<tr><td colspan="${headers.length + 1}" class="text-center">${noRecordsMessage}</td></tr>`;

        return `
            <div class="table-responsive">
                <table class="table table-bordered table-striped" ${tableId ? `id="${tableId}"` : ''}>
                    <thead class="table-secondary">
                        <tr>
                            ${headers.map(header => `<th>${header}</th>`).join('')}
                        </tr>
                    </thead>
                    <tbody>
                        ${tableContent}
                    </tbody>
                </table>
            </div>
        `;
    }

    renderPagination(currentPage, totalPages, type, pageSizeSelectId, currentPageSize) {
        let pageLinks = '';
        let startPage = Math.max(1, currentPage - 2);
        let endPage = Math.min(totalPages, currentPage + 2);

        if (totalPages <= 5) {

```

```

        startPage = 1;
        endPage = totalPages;
    } else {
        if (currentPage <= 3) {
            startPage = 1;
            endPage = 5;
        } else if (currentPage + 2 > totalPages) {
            startPage = totalPages - 4;
            endPage = totalPages;
        }
    }

    if (startPage > 1) {
        pageLinks += `<li class="page-item"><a class="page-link" href="#" data-page="${type}-1">1</a>`;
        if (startPage > 2) {
            pageLinks += `<li class="page-item disabled"><span class="page-link">...</span></li>`;
        }
    }

    for (let i = startPage; i <= endPage; i++) {
        pageLinks += `
        <li class="page-item ${i === currentPage ? 'active' : ''}">
            <a class="page-link" href="#" data-page="${type}-${i}">${i}</a>
        </li>`;
    }

    if (endPage < totalPages) {
        if (endPage < totalPages - 1) {
            pageLinks += `<li class="page-item disabled"><span class="page-link">...</span></li>`;
        }
        pageLinks += `<li class="page-item"><a class="page-link" href="#" data-page="${type}-${totalPages}">${totalPages}</a>`;
    }

    const pageSizeSelectOptions = estadoGlobal.PAGE_SIZE_OPTIONS.map(size =>
        `<option value="${size}" ${size === currentPageSize ? 'selected' : ''}>${size}</option>`
    ).join('');

    return `
    <nav aria-label="Page navigation" id="${type}-pagination-nav">
        <ul class="pagination justify-content-center">
            <li class="page-item ${currentPage === 1 ? 'disabled' : ''}">
                <a class="page-link" href="#" data-action="${type}-prev">Anterior</a>
            </li>
            ${pageLinks}
            <li class="page-item ${currentPage === totalPages || totalPages === 0 ? 'disabled' : ''}">
                <a class="page-link" href="#" data-action="${type}-next">Próximo</a>
            </li>
        </ul>
    </nav>
    <div class="d-flex justify-content-center my-2">
        <label class="me-2 align-self-center">Itens por página: </label>
        <select class="form-select w-auto" id="${pageSizeSelectId}" data-action="${type}-pagesize">
            ${pageSizeSelectOptions}
        </select>
    </div>
    `;
}

renderPageSizeSelect(id, currentPageSize) {
    const options = [5, 10, 25, 50, 100].map(size =>
        `<option value="${size}" ${currentPageSize === size ? 'selected' : ''}>${size}</option>`
    ).join('');
    return `
    <select class="form-select w-auto" id="${id}" data-action="${type}-pagesize">
        ${options}
    </select>
    `;
}

```

```

<div class="d-flex justify-content-center my-2">
  <label class="me-2 align-self-center">Itens por página: </label>
  <select class="form-select w-auto" id="{id}">
    {options}
  </select>
</div>
`;
}

// Abre o modal de detalhes do item, exibindo atributos padrão e controle de estoque apenas para tip
openItemDetail(item, qtdRetirada) {
  const userRole = getUserTypeFromToken();
  console.log(userRole);

  // Preenche campos comuns
  document.getElementById('itemNome').textContent = item.nome_item_original;
  document.getElementById('itemValidade').textContent = item.data_validade_item ? formatDate(item.

  // Controle de exibição de estoque (somente para almoxarifado)
  const liItemEstoque = document.getElementById('liItemEstoque');
  const liItemEstoqueMin = document.getElementById('liItemEstoqueMin');
  const itemEstoqueElement = document.getElementById('itemEstoque');
  const itemEstoqueMinElement = document.getElementById('itemEstoqueMin');
  if (userRole === 2 || userRole === 3) {
    if (liItemEstoque) {
      liItemEstoque.style.display = 'block';
      if (itemEstoqueElement) itemEstoqueElement.textContent = item.quantidade_item;
    }
    if (liItemEstoqueMin) {
      liItemEstoqueMin.style.display = 'block';
      if (itemEstoqueMinElement) {
        itemEstoqueMinElement.textContent =
          item.quantidade_minima_item !== null ? item.quantidade_minima_item : 'N/D';
      }
    }
  } else {
    if (liItemEstoque) liItemEstoque.style.display = 'none';
    if (liItemEstoqueMin) liItemEstoqueMin.style.display = 'none';
  }

  // Controle do campo "Solicitado"
  const itemQtdRetiradaElement = document.getElementById('itemQtdRetirada');
  const itemQtdRetiradaLi = itemQtdRetiradaElement ? itemQtdRetiradaElement.closest('li') : null;
  if (qtdRetirada !== undefined && qtdRetirada !== null) {
    itemQtdRetiradaElement.textContent = qtdRetirada;
    if (itemQtdRetiradaLi) itemQtdRetiradaLi.style.display = 'block';
  } else {
    if (itemQtdRetiradaLi) itemQtdRetiradaLi.style.display = 'none';
  }

  this.getModalInstance('modalDetalheItem').show();
}

fillModalDetalhes(retirada) {
  document.getElementById('detalheRetiradaId').value = retirada.retirada_id;
  document.getElementById('detalheStatus').value = getStatusText(retirada.status);
  document.getElementById('detalheSetor').value = retirada.setor_nome || '-';
  document.getElementById('detalheUsuario').value = retirada.usuario_nome || '-';
  document.getElementById('detalheSolicitadoPor').value = retirada.solicitado_localmente_por || '-';
  document.getElementById('detalheAutorizadoPor').value = retirada.autorizado_por_nome || '-';
  document.getElementById('detalheData').value = formatDateTime(retirada.data_solicitacao);
  document.getElementById('detalheJustificativa').value = retirada.justificativa || '-';
  document.getElementById('detalheStatusDesc').value = retirada.detalhe_status || '-';
}

```

```

        // Passa somente itens; openItemDetail cuida da exibição baseada em userRole
        this.renderItemList('detalheItens', retirada.itens);
    }

    fillModalDetalhesItem(item) {
        if (!item) {
            console.error("Item inválido para preencher detalhes do modal.");
            return;
        }
        // Reaproveita openItemDetail para preencher e exibir modal
        this.openItemDetail(item, null);
    }

    renderItemList(containerId, items, isRestrictedView = false) {
        const cont = document.getElementById(containerId);
        if (!cont) {
            console.error(`Container com ID ${containerId} não encontrado.`);
            return;
        }
        cont.innerHTML = '';
        items.forEach(i => {
            const btn = document.createElement('button');
            btn.type = 'button';
            btn.className = 'list-group-item list-group-item-action';
            btn.textContent = `${i.item.nome_item_original} Quantidade: ${i.quantidade_retirada}`;
            btn.onclick = () => this.openItemDetail(i.item, i.quantidade_retirada);
            cont.appendChild(btn);
        });
    }

    fillModalAutorizar(retirada) {
        document.getElementById('autorizarRetiradaId').value = retirada.retirada_id;
        document.getElementById('autorizarSetor').value = retirada.setor_nome || '-';
        document.getElementById('autorizarUsuario').value = retirada.usuario_nome || '-';
        document.getElementById('autorizarJustificativa').value = retirada.justificativa || '-';
        document.getElementById('autorizarData').value = formatDate(retirada.data_solicitacao);
        this.renderItemList('autorizarItens', retirada.itens);
        const btnConfirmarAutorizar = document.getElementById('btn-confirmar-autorizar-retirada');
        if (btnConfirmarAutorizar) btnConfirmarAutorizar.dataset.id = retirada.retirada_id;
        const btnConfirmarNegar = document.getElementById('btn-confirmar-negar-retirada');
        if (btnConfirmarNegar) btnConfirmarNegar.dataset.id = retirada.retirada_id;
    }

    fillModalConcluir(retirada) {
        document.getElementById('concluirRetiradaId').value = retirada.retirada_id;
        document.getElementById('concluirRetiradaDisplayId').textContent = retirada.retirada_id;
        document.getElementById('concluirDetalheStatus').value = retirada.detalhe_status || '';
    }

    getModalInstance(id) {
        const modalElement = document.getElementById(id);
        if (!modalElement) {
            console.error(`Elemento do modal com ID '${id}' não encontrado para getModalInstance.`);
            return null;
        }
        return bootstrap.Modal.getInstance(modalElement) || new bootstrap.Modal(modalElement);
    }

    showLoading() {
        if (this.loadingSpinner) {
            this.loadingSpinner.style.display = 'block';
        }
    }
}

```

```

hideLoading() {
  if (this.loadingSpinner) {
    this.loadingSpinner.style.display = 'none';
  }
}
}

```

```

export const uiService = new UiService();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\usuariosModule.js

```

// frontend/static/js/usuariosModule.js

import { apiService } from './apiService.js';
import { uiService } from './uiService.js';
import { showAlert } from './utils.js';

class UsuariosModule {
  constructor() {
    this.currentPage = 1;
    this.pageSize = 10;
    this.searchNome = ''; // Current search term for user name
    this.totalUsers = 0; // To store total users for pagination calculation
    this.setorIdToNameMap = {}; // Novo mapa para armazenar ID do setor -> Nome do setor

    // Modals
    this.modalCadastrarUsuario = uiService.getModalInstance('modalCadastrarUsuario');
    this.modalEditarUsuario = uiService.getModalInstance('modalEditarUsuario');
    this.modalConfirmarDeleteUsuario = uiService.getModalInstance('modalConfirmarDeleteUsuario');

    // Forms
    this.formCadastrarUsuario = document.getElementById('form-cadastrar-usuario');
    this.formEditarUsuario = document.getElementById('form-editar-usuario');

    // Buttons
    this.btnSalvarCadastrarUsuario = document.getElementById('btn-salvar-cadastrar-usuario');
    this.btnSalvarEditarUsuario = document.getElementById('btn-salvar-editar-usuario');
    this.btnConfirmarDeletarUsuario = document.getElementById('btn-confirmar-deletar-usuario');

    // Bindings
    this.boundHandlePaginationClick = this._handlePaginationClick.bind(this);
    this.boundHandlePageSizeChange = this._handlePageSizeChange.bind(this);
    this.boundHandleSearchUsers = this._handleSearchUsers.bind(this);
    this.boundHandleClearSearch = this._handleClearSearch.bind(this);
    this.boundHandleTableActions = this._handleTableActions.bind(this);
    this.boundUpdateUsuario = this._updateUsuario.bind(this); // Bind update method
  }

  init() {
    this.bindEvents();
  }

  bindEvents() {
    // Cadastrar Usuário
    document.getElementById('btn-open-cadastrar-usuario')?.addEventListener('click', e => {
      e.preventDefault();
      this.formCadastrarUsuario.reset();
      this.populateSetoresInForm(this.formCadastrarUsuario.querySelector('select[name="setor_id"]'));
      this.modalCadastrarUsuario.show();
    });
  }
}

```

```

    this.btnSalvarCadastrarUsuario?.addEventListener('click', () => this._createUsuario());

    // Confirmar delete
    this.btnConfirmarDeletarUsuario?.addEventListener('click', this._deleteUsuarioConfirmed.bind(this));

    // Salvar Editar Usuario
    this.btnSalvarEditarUsuario?.addEventListener('click', this.boundUpdateUsuario);
  }

  async populateSetoresInForm(selectElement, selectedSetorId = null) {
    selectElement.innerHTML = '<option value="" disabled selected>Carregando setores...</option>';
    try {
      const setores = await apiService.fetchAllSetores();
      selectElement.innerHTML = '<option value="" disabled selected>Selecione um setor</option>';
      setores.forEach(setor => {
        const option = document.createElement('option');
        option.value = setor.setor_id;
        option.textContent = setor.nome_setor;
        if (selectedSetorId !== null && setor.setor_id === selectedSetorId) {
          option.selected = true;
        }
        selectElement.appendChild(option);
      });
    } catch (error) {
      console.error('Erro ao carregar setores:', error);
      showAlert('Erro ao carregar setores.', 'danger');
      selectElement.innerHTML = '<option value="" disabled selected>Erro ao carregar</option>';
    }
  }

  async renderUsuariosList() {
    uiService.showLoading();
    try {
      // Fetch all users. The backend endpoint /usuarios returns all users.
      const allUsers = await apiService.get('/usuarios');

      // Fetch all sectors to map IDs to names
      const allSetores = await apiService.get('/setores');
      this.setorIdToNameMap = {};
      allSetores.forEach(setor => {
        this.setorIdToNameMap[setor.setor_id] = setor.nome_setor;
      });

      // Frontend filtering and pagination for now
      const filteredUsers = this.searchNome
        ? allUsers.filter(u => u.nome_usuario.toLowerCase().includes(this.searchNome.toLowerCase()))
        : allUsers;

      this.totalUsers = filteredUsers.length; // Update total users count
      const totalPages = Math.ceil(this.totalUsers / this.pageSize);

      const offset = (this.currentPage - 1) * this.pageSize;
      const usersForPage = filteredUsers.slice(offset, offset + this.pageSize);

      const tableHeaders = ['ID', 'Nome', 'Email', 'Tipo', 'Setor', 'SIAPE', 'Ações'];
      const tableHtml = uiService.renderTable(tableHeaders, usersForPage, {
        noRecordsMessage: "Nenhum usuário encontrado.",
        rowMapper: (usuario) => {
          // Use the map to get the sector name
          const sectorName = this.setorIdToNameMap[usuario.setor_id] || 'N/D';
          const userRoleText = this._getRoleText(usuario.tipo_usuario);
          return [
            usuario.usuario_id,

```



```

        usuario.nome_usuario,
        usuario.email_usuario,
        userRoleText,
        sectorName, // Exibe o nome do setor
        usuario.siape_usuario || 'N/D'
    ];
},
actionsHtml: (usuario) => `
    <div class="d-flex flex-wrap justify-content-center gap-1">
        <button class="btn btn-sm btn-primary btn-acoes btn-editar-usuario" data-id="${u
            <i class="bi bi-pencil-square"></i> Editar
        </button>
        <button class="btn btn-sm btn-danger btn-acoes btn-deletar-usuario" data-id="${u
            <i class="bi bi-trash"></i> Deletar
        </button>
    </div>
`
    );

const paginationHtml = uiService.renderPagination(
    this.currentPage,
    totalPages,
    'usuarios', // type for pagination links
    'usuariosPageSizeSelect', // id for pageSize select element
    this.pageSize
);

const searchBarHtml = `
    <div class="card mb-3">
        <div class="card-header">Filtros de Busca</div>
        <div class="card-body">
            <form id="search-bar-usuarios" class="row g-3 mb-0">
                <div class="col-12 col-md">
                    <label for="search-usuario-nome" class="form-label">Nome do Usuário</lab
                    <input type="text" id="search-usuario-nome" class="form-control" placeho
                </div>
                <div class="col-12 col-md d-flex justify-content-end align-items-end">
                    <button type="button" id="btn-search-usuario" class="btn btn-primary me-
                    <button type="button" id="btn-clear-search-usuario" class="btn btn-secon
                </div>
            </form>
        </div>
    </div>
`;

uiService.renderPage(
    'Gerenciamento de Usuários',
    `${searchBarHtml}${tableHtml}${paginationHtml}`
);

this._bindPageEvents(); // Corrigido: Chamada do método interno
this._bindTableActions(); // Corrigido: Chamada do método interno

} catch (error) {
    console.error('Erro ao renderizar lista de usuários:', error);
    showAlert(error.message || 'Erro ao carregar usuários.', 'danger');
    uiService.renderPage('Gerenciamento de Usuários', `<div class="alert alert-warning">Erro ao
} finally {
    uiService.hideLoading();
}

}

_getRoleText(roleValue) {

```

```

// Based on RoleEnum from backend: 1: Geral, 2: Almoxarifado, 3: Direcao
switch (roleValue) {
  case 1: return '<span class="badge bg-secondary">Geral</span>';
  case 2: return '<span class="badge bg-info">Almoxarifado</span>';
  case 3: return '<span class="badge bg-primary">Direção</span>';
  default: return 'Desconhecido';
}
}

_bindPageEvents() {
  // Select elements by ID. If they are part of dynamic content, ensure they exist.
  const paginationNav = document.getElementById('usuarios-pagination-nav');
  const pageSizeSelect = document.getElementById('usuariosPageSizeSelect');
  const btnSearch = document.getElementById('btn-search-usuario');
  const btnClearSearch = document.getElementById('btn-clear-search-usuario');

  // Removendo listeners antigos para evitar duplicação
  if (paginationNav) {
    paginationNav.removeEventListener('click', this.boundHandlePaginationClick);
  }
  if (pageSizeSelect) {
    pageSizeSelect.removeEventListener('change', this.boundHandlePageSizeChange);
  }
  if (btnSearch) {
    btnSearch.removeEventListener('click', this.boundHandleSearchUsers);
  }
  if (btnClearSearch) {
    btnClearSearch.removeEventListener('click', this.boundHandleClearSearch);
  }

  // Adicionando novos listeners
  if (paginationNav) {
    paginationNav.addEventListener('click', this.boundHandlePaginationClick);
  }
  if (pageSizeSelect) {
    pageSizeSelect.addEventListener('change', this.boundHandlePageSizeChange);
  }
  if (btnSearch) {
    btnSearch.addEventListener('click', this.boundHandleSearchUsers);
  }
  if (btnClearSearch) {
    btnClearSearch.addEventListener('click', this.boundHandleClearSearch);
  }
}

_handlePaginationClick(e) {
  e.preventDefault();
  const clickedPageLink = e.target.closest('a[data-page^="usuarios-"]');
  const clickedActionButton = e.target.closest('a[data-action^="usuarios-"]');

  if (clickedPageLink) {
    const pageValue = clickedPageLink.dataset.page.split('-')[1];
    const newPage = parseInt(pageValue);
    const totalPages = Math.ceil(this.totalUsers / this.pageSize); // Cálculo de totalPages
    if (!isNaN(newPage) && newPage !== this.currentPage && newPage >= 1 && newPage <= totalPages) {
      this.currentPage = newPage;
      this.renderUsuariosList();
    }
    return;
  }

  if (clickedActionButton) {
    const action = clickedActionButton.dataset.action;

```

```

        let newPage = this.currentPage;
        const totalPages = Math.ceil(this.totalUsers / this.pageSize); // Cálculo de totalPages

        if (action === 'usuarios-prev' && newPage > 1) {
            newPage--;
        } else if (action === 'usuarios-next' && newPage < totalPages) {
            newPage++;
        }

        if (newPage !== this.currentPage) {
            this.currentPage = newPage;
            this.renderUsuariosList();
        }
        return;
    }
}

_handlePageSizeChange(e) {
    this.pageSize = parseInt(e.target.value);
    this.currentPage = 1; // Resetar para primeira página quando muda o tamanho da página
    this.renderUsuariosList();
}

_handleSearchUsers(e) {
    e.preventDefault();
    this.searchNome = document.getElementById('search-usuario-nome').value.trim();
    this.currentPage = 1; // Resetar para primeira página quando houver busca por filtro
    this.renderUsuariosList();
}

_handleClearSearch(e) {
    e.preventDefault();
    document.getElementById('search-usuario-nome').value = '';
    this.searchNome = '';
    this.currentPage = 1; // Resetar para primeira página quando limpar os filtros de busca
    this.renderUsuariosList();
}

_bindTableActions() {
    // Event delegation on main-content for buttons that are dynamically rendered
    const mainContent = document.getElementById('main-content');
    if (mainContent) {
        mainContent.removeEventListener('click', this.boundHandleTableActions); // Remove listener a
        mainContent.addEventListener('click', this.boundHandleTableActions); // Adiciona listener no
    }
}

async _handleTableActions(e) {
    const editButton = e.target.closest('.btn-editar-usuario');
    const deleteButton = e.target.closest('.btn-deletar-usuario');

    if (editButton) {
        const userId = parseInt(editButton.dataset.id);
        await this._openEditModal(userId);
    } else if (deleteButton) {
        const userId = parseInt(deleteButton.dataset.id);
        this._openDeleteConfirmModal(userId);
    }
}

async _createUsuario() {
    if (!this.formCadastrarUsuario.checkValidity()) {
        this.formCadastrarUsuario.reportValidity();
    }
}

```

```

        return;
    }

    const formData = new FormData(this.formCadastrarUsuario);
    const userData = {};
    for (const [key, value] of formData.entries()) {
        if (value !== '') { // Exclude empty fields
            userData[key] = value;
        }
    }

    // Convertendo tipos
    userData.tipo_usuario = parseInt(userData.tipo_usuario);
    userData.setor_id = parseInt(userData.setor_id);
    userData.siape_usuario = userData.siape_usuario ? userData.siape_usuario.trim() : null;

    uiService.showLoading();
    try {
        await apiService.post('/usuarios/', userData);
        showAlert('Usuário cadastrado com sucesso!', 'success');
        this.modalCadastrarUsuario.hide();
        this.renderUsuariosList(); // atualizar lista
    } catch (error) {
        console.error('Erro ao cadastrar usuário:', error);
        showAlert(error.message || 'Erro ao cadastrar usuário.', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

async _openEditModal(userId) {
    uiService.showLoading();
    try {
        const user = await apiService.get(`/usuarios/${userId}`);
        this.formEditarUsuario.querySelector('input[name="nome_usuario"]').value = user.nome_usuario;
        this.formEditarUsuario.querySelector('input[name="email_usuario"]').value = user.email_usuario;
        this.formEditarUsuario.querySelector('select[name="tipo_usuario"]').value = user.tipo_usuario;
        this.formEditarUsuario.querySelector('input[name="username"]').value = user.username;
        this.formEditarUsuario.querySelector('input[name="siape_usuario"]').value = user.siape_usuario;

        // Popular e selecionar setor atual
        const setorSelect = this.formEditarUsuario.querySelector('select[name="setor_id"]');
        await this.populateSetoresInForm(setorSelect, user.setor_id);

        // Armazenar user ID com clique no botão de salvar
        this.btnSalvarEditarUsuario.dataset.id = userId;
        this.modalEditarUsuario.show();
    } catch (error) {
        console.error('Erro ao carregar dados do usuário para edição', error);
        showAlert(error.message || 'Erro ao carregar dados do usuário.', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

async _updateUsuario() {
    const userId = parseInt(this.btnSalvarEditarUsuario.dataset.id);
    if (!userId) {
        showAlert('ID do usuário para edição não encontrado.', 'danger');
        return;
    }

    if (!this.formEditarUsuario.checkValidity()) {

```

```

        this.formEditarUsuario.reportValidity();
        return;
    }

    const formData = new FormData(this.formEditarUsuario);
    const userData = {};
    for (const [key, value] of formData.entries()) {
        // Only include fields that have a value (for optional fields like password)
        if (value !== '') {
            userData[key] = value;
        }
    }

    // Ensure numbers are parsed correctly, and undefined for optional unset fields
    userData.tipo_usuario = userData.tipo_usuario ? parseInt(userData.tipo_usuario) : undefined;
    userData.setor_id = userData.setor_id ? parseInt(userData.setor_id) : undefined;
    userData.siape_usuario = userData.siape_usuario ? userData.siape_usuario.trim() : null;

    uiService.showLoading();
    try {
        await apiService.put(`/usuarios/${userId}`, userData); // Use PUT for full update
        showAlert('Usuário atualizado com sucesso!', 'success');
        this.modalEditarUsuario.hide();
        this.renderUsuariosList(); // Refresh list
    } catch (error) {
        console.error('Erro ao atualizar usuário:', error);
        showAlert(error.message || 'Erro ao atualizar usuário.', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

_openDeleteConfirmModal(userId) {
    document.getElementById('confirm-delete-user-id').textContent = userId;
    this.btnConfirmarDeletarUsuario.dataset.id = userId; // Store ID for confirmation
    this.modalConfirmarDeleteUsuario.show();
}

async _deleteUsuarioConfirmed() {
    const userId = parseInt(this.btnConfirmarDeletarUsuario.dataset.id);
    if (!userId) {
        showAlert('ID do usuário para exclusão não encontrado.', 'danger');
        return;
    }

    uiService.showLoading();
    try {
        await apiService.delete(`/usuarios/${userId}`);
        showAlert('Usuário deletado com sucesso!', 'success');
        this.modalConfirmarDeleteUsuario.hide();
        this.renderUsuariosList(); // Refresh list
    } catch (error) {
        console.error('Erro ao deletar usuário', error);
        showAlert(error.message || 'Erro ao deletar usuário.', 'danger');
    } finally {
        uiService.hideLoading();
    }
}

}

export const usuariosModule = new UsuariosModule();

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\utils.js

```
// frontend/static/js/utils.js

import estadoGlobal from './estadoGlobal.js';

export function showAlert(message, type = 'success', duration = 5000) {
  const toastContainer = document.getElementById('toast-container');
  if (!toastContainer) {
    console.error("Elemento 'toast-container' não encontrado para exibir alerta.");
    return;
  }

  const toastElement = document.createElement('div');
  toastElement.className = `toast align-items-center text-white bg-${type} border-0 fade show`;
  toastElement.setAttribute('role', 'alert');
  toastElement.setAttribute('aria-live', 'assertive');
  toastElement.setAttribute('aria-atomic', 'true');
  toastElement.innerHTML = `
    <div class="d-flex">
      <div class="toast-body">
        ${message}
      </div>
      <button type="button" class="btn-close btn-close-white me-2 m-auto" data-bs-dismiss="toast"
    </div>
  `;

  toastContainer.appendChild(toastElement);

  const toast = new bootstrap.Toast(toastElement, {
    autohide: true,
    delay: duration
  });
  toast.show();

  toastElement.addEventListener('hidden.bs.toast', () => {
    toastElement.remove();
  });
}

export function formatDate(dateString) {
  if (!dateString) return "N/A";
  const date = new Date(dateString);
  return date.toLocaleDateString('pt-BR');
}

export function formatDateTime(dateString) {
  if (!dateString) return "N/A";
  const date = new Date(dateString);
  return date.toLocaleString('pt-BR');
}

export function getStatusText(statusCode) {
  return estadoGlobal.statusMap[statusCode] || 'Desconhecido';
}

export function getStatusValue(statusName) {
  return estadoGlobal.statusMapUpdate[statusName];
}

const NEW_ALERTS_FLAG_KEY = 'hasNewAlerts';
```

```

const NEW_WITHDRAWAL_REQUESTS_FLAG_KEY = 'hasNewWithdrawalRequests';

export function setNewAlertsFlag(hasNewAlerts) {
  if (hasNewAlerts) {
    localStorage.setItem(NEW_ALERTS_FLAG_KEY, 'true');
  } else {
    localStorage.removeItem(NEW_ALERTS_FLAG_KEY);
  }
  updateNotificationBellUI(); // Atualiza a UI imediatamente
}

export function getNewAlertsFlag() {
  return localStorage.getItem(NEW_ALERTS_FLAG_KEY) === 'true';
}

export function setNewWithdrawalRequestsFlag(hasNewRequests) {
  if (hasNewRequests) {
    localStorage.setItem(NEW_WITHDRAWAL_REQUESTS_FLAG_KEY, 'true');
  } else {
    localStorage.removeItem(NEW_WITHDRAWAL_REQUESTS_FLAG_KEY);
  }
  updateNotificationBellUI(); // Atualiza a UI imediatamente
}

export function getNewWithdrawalRequestsFlag() {
  return localStorage.getItem(NEW_WITHDRAWAL_REQUESTS_FLAG_KEY) === 'true';
}

export function updateNotificationBellUI() {
  const notificationDot = document.getElementById('notification-dot');
  const newAlerts = getNewAlertsFlag();
  const newWithdrawalRequests = getNewWithdrawalRequestsFlag();

  if (notificationDot) {
    if (newAlerts || newWithdrawalRequests) {
      notificationDot.style.display = 'block';
      notificationDot.classList.add('animate__animated', 'animate__pulse');
    } else {
      notificationDot.style.display = 'none';
      notificationDot.classList.remove('animate__animated', 'animate__pulse');
    }
  } else {
    console.warn('updateNotificationBellUI: Elemento #notification-dot não encontrado.');
  }

  const newAlertsMenuItem = document.getElementById('new-alerts-menu-item');
  const newWithdrawalRequestsMenuItem = document.getElementById('new-withdrawal-requests-menu-item');
  const noNotificationsMenuItem = document.getElementById('no-notifications-menu-item');

  if (newAlertsMenuItem) {
    newAlertsMenuItem.style.display = newAlerts ? 'block' : 'none';
  }

  if (newWithdrawalRequestsMenuItem) {
    newWithdrawalRequestsMenuItem.style.display = newWithdrawalRequests ? 'block' : 'none';
  }

  // Lógica para exibir/ocultar "Sem notificações"
  if (noNotificationsMenuItem) {
    if (!newAlerts && !newWithdrawalRequests) { // Corrigido para verificar ambos os flags
      noNotificationsMenuItem.style.display = 'block'; // Exibe "Sem notificações"
    } else {
      noNotificationsMenuItem.style.display = 'none'; // Oculta "Sem notificações"
    }
  }
}

```

```

    }
  }
}

// Função para decodificar o token JWT e obter o ID do usuário
export function getUserIdFromToken() {
  const token = localStorage.getItem('token');
  if (!token) {
    return null;
  }
  try {
    const payload = JSON.parse(atob(token.split('.')[1]));
    return payload.usuario_id || null; // Retorna o ID do usuário
  } catch (e) {
    console.error("Erro ao decodificar token JWT:", e);
    return null;
  }
}

export function getUserTypeFromToken() {
  const token = localStorage.getItem('token');
  if (!token) {
    return null;
  }
  try {
    const payload = JSON.parse(atob(token.split('.')[1]));
    return payload.tipo_usuario || null;
  } catch (e) {
    console.error("Erro ao decodificar token JWT:", e);
    return null;
  }
}

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\static\js\validar-acesso.js

```

//frontend\static\js\validar-acesso.js

document.addEventListener('DOMContentLoaded', function() {

  document.getElementById('form-login').addEventListener('submit', async function (event) {
    event.preventDefault();

    const form = event.target;
    const formData = new FormData(form);
    const data = new URLSearchParams(formData);

    try {
      const response = await fetch('/api/almoxarifado/usuarios/token', {
        method: 'POST',
        headers: {
          'Accept': 'application/json'
        },
        body: data,
        credentials: 'include' // permite que o cookie seja salvo
      });

      if (response.ok) {
        // Depois de salvar o cookie com sucesso, redireciona manualmente
        window.location.href = "/";
      } else {

```



```

        const result = await response.json();
        alert(result.detail || "Erro ao fazer login");
    }
    } catch (err) {
        console.error("Erro no login:", err);
        alert("Erro de rede ao tentar fazer login.");
    }
    });
});
});

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\dashboardAlmoxarifado.html

```

<!-- frontend/templates/dashboardAlmoxarifado.html -->
{% include 'partials/head.html' %}
<link rel="stylesheet" href="/static/css/dashboards.css" />
<link rel="stylesheet" href="/static/css/custom.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">
</head>
<body>

{% include 'partials/topbar.html' %}
{% include 'partials/navbar.html' %}

<!-- Conteúdo principal dinâmico -->
<div class="container mt-5" id="main-content">
    <!-- Seção de Boas-Vindas Personalizada para Almoxarifado -->
    <h3 class="mb-3">Olá, <span id="welcome-user-name-almoxarifado">Almoxarifado</span>!</h3>
    <p class="text-muted">Bem-vindo(a) ao Sistema de Controle de Almoxarifado do CPT.</p>

    <div class="card bg-white rounded-lg shadow-sm p-4 mb-4">
        <p class="mb-1"><strong>SIAPE:</strong> <span id="user-siape-almoxarifado">Carregando...</span></p>
        <p class="mb-0"><strong>Setor:</strong> <span id="user-sector-almoxarifado">Carregando...</span></p>
    </div>

    {# Inclui os cards de acesso rápido existentes via partial #}
    {% include 'partials/quick_access.html' %}
</div>

<!-- Partial: global_overlays.html -->
{% include 'partials/global_overlays.html' %}

<!-- Incluindo modais de Itens -->
{% include 'partials/modals/modal_cadastrar_item.html' %}
{% include 'partials/modals/modal_editar_item.html' %}

<!-- Incluindo modais de Categorias -->
{% include 'partials/modals/modal_cadastrar_categoria.html' %}
{% include 'partials/modals/modal_editar_categoria.html' %}

<!-- Incluindo modais de Retiradas -->
{% include 'partials/modals/modal_ver_detalhes_retirada.html' %}
{% include 'partials/modals/modal_autorizar_retirada.html' %}
{% include 'partials/modals/modal_item_detalhes.html' %}
{% include 'partials/modals/modal_solicitar_retirada.html' %}
{% include 'partials/modals/modal_selecionar_item_retirada.html' %}
{% include 'partials/modals/modal_concluir_retirada.html' %}

<!-- Incluindo modais de Relatórios -->
{% include 'partials/modals/modal_reports_dashboard.html' %}
{% include 'partials/modals/modal_report_quantidade_itens.html' %}

```

```

{% include 'partials/modals/modal_report_entrada_itens.html' %}
{% include 'partials/modals/modal_report_retiradas_setor.html' %}
{% include 'partials/modals/modal_report_retiradas_usuario.html' %}

<div class="modal fade" id="modalImportarTabelaltens" tabindex="-1" aria-labelledby="modalImportarTabela
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalImportarTabelaltensLabel">Importar Tabela de Itens</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
      </div>
      <div class="modal-body">
        <form id="form-importar-tabela-itens">
          <div class="mb-3">
            <label for="arquivoltens" class="form-label">Selecione o arquivo (.xlsx ou .csv)</label>
            <input class="form-control" type="file" id="arquivoltens" name="file" accept=".xlsx, .csv">
          </div>
        </form>
        <div id="import-feedback" class="mt-3" style="display:none;">
          <div class="alert" role="alert" id="import-alert"></div>
          <ul id="import-errors-list" class="list-group list-group-flush"></ul>
        </div>
      </div>
      <div class="modal-footer border-0 justify-content-center">
        <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
        <button type="button" class="btn btn-primary" id="btn-enviar-tabela-itens">Enviar</button>
      </div>
    </div>
  </div>
</div>

{% include 'partials/scripts.html' %}

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\dashboardDirecao.html

```

<!-- frontend/templates/dashboardDirecao.html -->
{% include 'partials/head.html' %}
<link rel="stylesheet" href="/static/css/dashboards.css" />
<link rel="stylesheet" href="/static/css/custom.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">
</head>
<body>

  {% include 'partials/topbar.html' %}
  {% include 'partials/navbar_direcao.html' %}

  <!-- Conteúdo principal dinâmico -->
  <div class="container mt-5" id="main-content">
    <!-- Seção de Boas-Vindas Personalizada para Direção -->
    <h3 class="mb-3">Olá, <span id="welcome-user-name-direcao">Direção</span>!</h3>
    <p class="text-muted">Bem-vindo(a) ao Sistema de Controle de Almoxarifado do CPT.</p>

    <div class="card bg-white rounded-lg shadow-sm p-4 mb-4">
      <p class="mb-1"><strong>SIAPE:</strong> <span id="user-siape-direcao">Carregando...</span></p>
      <p class="mb-0"><strong>Setor:</strong> <span id="user-sector-direcao">Carregando...</span></p>
    </div>

    <h3 class="mb-3 mt-5">Acesso Rápido</h3>
    <div class="row g-3">
      {# Cartões de acesso rápido específicos para a Direção #}
    </div>
  </div>

```

```

<div class="col-6 col-md-3">
  <a href="#" id="listar-usuarios-link-quick" class="text-decoration-none">
    <div class="quick-access-card">
      <i class="bi bi-person-fill-gear"></i>
      <p>Lista de Usuários</p>
    </div>
  </a>
</div>
<div class="col-6 col-md-3">
  <a href="#" id="listar-setores-link-quick" class="text-decoration-none">
    <div class="quick-access-card">
      <i class="bi bi-building-fill"></i>
      <p>Lista de Setores</p>
    </div>
  </a>
</div>
<div class="col-6 col-md-3">
  <a href="#" id="open-alertas-modal-quick" class="text-decoration-none">
    <div class="quick-access-card">
      <i class="bi bi-exclamation-octagon-fill"></i>
      <p>Histórico de Alertas</p>
    </div>
  </a>
</div>
<div class="col-6 col-md-3">
  <a href="#" id="open-reports-dashboard-quick" data-bs-toggle="modal" data-bs-target="#modalR
    <div class="quick-access-card">
      <i class="bi bi-bar-chart-line-fill"></i>
      <p>Modal de Relatórios</p>
    </div>
  </a>
</div>
</div>
</div>

<!-- Partial: global_overlays.html -->
{% include 'partials/global_overlays.html' %}

<!-- Incluindo modais de Usuários -->
{% include 'partials/modals/modal_cadastrar_usuario.html' %}
{% include 'partials/modals/modal_editar_usuario.html' %}
{% include 'partials/modals/modal_confirmar_deletar_usuario.html' %}

<!-- NOVO: Incluindo modais de Setores -->
{% include 'partials/modals/modal_cadastrar_setor.html' %}
{% include 'partials/modals/modal_editar_setor.html' %}
{% include 'partials/modals/modal_confirmar_deletar_setor.html' %}

<!-- Incluindo modais de Retiradas -->
{% include 'partials/modals/modal_ver_detalhes_retirada.html' %}
{% include 'partials/modals/modal_autorizar_retirada.html' %}
{% include 'partials/modals/modal_solicitar_retirada.html' %}
{% include 'partials/modals/modal_selecionar_item_retirada.html' %}
{% include 'partials/modals/modal_concluir_retirada.html' %}

<!-- Incluindo modais de Itens (detalhes) -->
{% include 'partials/modals/modal_item_detalhes.html' %}

<!-- Incluindo modais de Relatórios -->
{% include 'partials/modals/modal_reports_dashboard.html' %}
{% include 'partials/modals/modal_report_quantidade_itens.html' %}
{% include 'partials/modals/modal_report_entrada_itens.html' %}
{% include 'partials/modals/modal_report_retiradas_setor.html' %}

```

```

{% include 'partials/modals/modal_report_retiradas_usuario.html' %}

{% include 'partials/scripts.html' %}

<script type="module" src="/static/js/usuariosModule.js"></script>
<script type="module" src="/static/js/setoresModule.js"></script> {# NOVO SCRIPT #}
<script type="module" src="/static/js/main_direcao.js"></script>

</body>
</html>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\dashboardServidor.html

```

<!DOCTYPE html>
<html lang="pt-br">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Sistema de Controle de Almoxarifado Dashboard Servidor</title>

  <!-- Bootstrap CSS offline -->
  <link href="/static/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet">
  <link rel="stylesheet" href="/static/css/dashboards.css" />
  <link rel="stylesheet" href="/static/css/custom.css">
  <!-- Animate.css para animações (usado no sino de notificação) -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">
</head>

<body>

  <!-- navbar e topbar -->
  {% include 'partials/topbar.html' %}
  {% include 'partials/navbar_servidor.html' %}

  <!-- Conteúdo principal dinâmico -->
  <div class="container mt-5" id="main-content">
    <!-- Seção de Boas-Vindas Personalizada -->
    <h3 class="mb-3">Olá, <span id="welcome-user-name">Servidor(a)</span>!</h3>
    <p class="text-muted">Bem-vindo(a) ao Sistema de Controle de Almoxarifado do CPT.</p>
    <div class="card bg-white rounded-lg shadow-sm p-4 mb-4">
      <p class="mb-1"><strong>SIAPE:</strong> <span id="user-siape">Carregando...</span></p>
      <p class="mb-0"><strong>Setor:</strong> <span id="user-sector">Carregando...</span></p>
    </div>

    <!-- Seção de Últimas Retiradas com Barra de Progresso -->
    <h4 class="mb-3 mt-5">Minhas Últimas Solicitações</h4>
    <div class="card bg-white rounded-lg shadow-sm p-4">
      <div id="latest-withdrawals-container" class="list-group">
        <!-- Conteúdo dinâmico carregado via JS -->
        <p class="text-muted text-center" id="loading-recent-withdrawals">Carregando últimas solicitações</p>
        <p class="text-muted text-center" id="no-recent-withdrawals" style="display: none;">Nenhuma solicitação encontrada</p>
      </div>
      <div class="text-center mt-3">
        <a href="#" id="view-all-my-withdrawals-link" class="btn btn-sm btn-outline-primary">Ver todas</a>
      </div>
    </div>
  </div>

  <!-- Partial: global_overlays.html -->

```

```

    {% include 'partials/global_overlays.html' %}

    <!-- Incluindo modais -->
    {% include 'partials/modals/modal_ver_detalhes_retirada.html' %}
    {% include 'partials/modals/modal_autorizar_retirada.html' %}
    {% include 'partials/modals/modal_item_detalhes.html' %}
    {% include 'partials/modals/modal_solicitar_retirada.html' %}
    {% include 'partials/modals/modal_selecionar_item_retirada.html' %}

    <!-- Partial: scripts.html -->
    <script src="/static/js/bootstrap.bundle.min.js"></script>
    <script src="/static/js/logout.js"></script>
    <script type="module" src="/static/js/estadoGlobal.js"></script>
    <script type="module" src="/static/js/utils.js"></script>
    <script type="module" src="/static/js/apiService.js"></script>
    <script type="module" src="/static/js/dataService.js"></script>
    <script type="module" src="/static/js/uiService.js"></script>
    <script type="module" src="/static/js/solicitar-retirada.js"></script>
    <script type="module" src="/static/js/selecionar-item-module.js"></script>
    <script type="module" src="/static/js/historicoServidorModule.js"></script>
    <script type="module" src="/static/js/main.js"></script>

</body>

</html>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxa rifado_ets\frontend\templates\index.html

```

    <!-- frontend\templates\index.html -->

    <!DOCTYPE html>
    <html lang="pt-BR">
    <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Login - Sistema Almoxarifado</title>
        <link rel="stylesheet" href="/static/css/style.css" />
    </head>
    <body>
        <header>
            <h2>Sistema de Controle de Almoxarifado - CPT</h2>
        </header>
        <main>
            <div class="login-card">
                <div class="avatar-placeholder"></div>
                <form id="login-form">
                    <label for="username">Usuário:</label>
                    <input type="text" id="username" name="username" required />

                    <label for="password">Senha:</label>
                    <input type="password" id="password" name="password" required />

                    <a href="#" id="recover-link">Cadastrar/Recuperação de conta</a>

                    <button type="submit" id="login-btn">Entrar</button>
                </form>
            </div>
        </main>
    <script src="/static/js/login.js"></script>

```

```

<script src="/static/js/validar-acesso.js"></script> </body>

</body>
</html>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\global_overlays.html

```

<!-- frontend\templates\partials\global_overlays.html -->

<div id="loading-spinner" style="
    display: none;
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 1060; /* Acima de modais do Bootstrap (z-index padrão do modal é 1050) */
    background-color: rgba(255, 255, 255, 0.7); /* Fundo semi-transparente */
    border-radius: 10px;
    padding: 20px;
    backdrop-filter: blur(2px); /* Adiciona um leve desfoque ao fundo */
    -webkit-backdrop-filter: blur(2px); /* Para compatibilidade com navegadores WebKit */
">
    <div class="spinner-border text-primary" role="status" style="width: 3rem; height: 3rem;">
        <span class="visually-hidden">Carregando...</span>
    </div>
    <p class="mt-2 text-primary">Carregando...</p>
</div>

<div id="toast-container" class="toast-container position-fixed top-0 end-0 p-3" style="z-index: 1070;">
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\head.html

```

<!-- frontend\templates\partials\head.html -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Sistema de Controle de Almoxarifado</title>

    <link href="/static/css/bootstrap.min.css" rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet">
    <link rel="stylesheet" href="/static/css/dashboards.css" />
    <link rel="stylesheet" href="/static/css/custom.css"> </head>
<body>
    <script src="/static/js/validar-acesso.js"></script>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\navbar.html

```

<nav class="navbar navbar-expand-lg navbar-dark navbar-custom">
    <div class="container-fluid">

```

```

<button class="btn btn-outline-light me-3" id="home-button"><i class="bi bi-house-fill"></i></button>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarMenu"
    aria-controls="navbarMenu" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="navbarMenu">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        {% include 'partials/nav_items.html' %}
    </ul>

    <div class="d-flex align-items-center">
        <div class="dropdown">
            <a class="nav-link position-relative me-3" href="#" id="alert-notification-bell" role="button">
                <i class="bi bi-bell-fill fs-5"></i>
                <span id="notification-dot" class="notification-dot position-absolute">
                    <span class="visually-hidden">Novas notificações</span>
                </span>
            </a>
            <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="alert-notification-bell">
                <li><span class="dropdown-item text-muted" id="no-notifications-menu-item" style="display: block; text-align: center; padding: 5px 0;">Nenhuma notificação</span>
                <li><a class="dropdown-item" href="#" id="new-withdrawal-requests-menu-item" style="display: block; text-align: center; padding: 5px 0;">Novas solicitações de saque</a>
                <li><a class="dropdown-item" href="#" id="new-alerts-menu-item" style="display: block; text-align: center; padding: 5px 0;">Novas alertas</a>
                <li><hr class="dropdown-divider"></li>
            </ul>
        </div>

        <div class="dropdown">
            <a class="nav-link dropdown-toggle" href="#" id="userMenu" data-bs-toggle="dropdown"
                aria-expanded="false">
                <i class="bi bi-person-circle"></i>
            </a>
            <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="userMenu">
                <li><a class="dropdown-item" href="#" id="logout-link">Sair</a></li>
            </ul>
        </div>
    </div>
</div>
</div>
</nav>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\navbar_direcao.html

```

<!-- frontend/templates/partials/navbar_direcao.html -->
<nav class="navbar navbar-expand-lg navbar-dark navbar-custom">
    <div class="container-fluid">
        <button class="btn btn-outline-light me-3" id="home-button"><i class="bi bi-house-fill"></i></button>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarMenu"
            aria-controls="navbarMenu" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarMenu">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <!-- Usuários -->
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#" id="usuariosDropdown" data-bs-toggle="dropdown">
                        <i class="bi bi-people"></i>
                    </a>
                    <ul class="dropdown-menu" aria-labelledby="usuariosDropdown">
                        <li><a class="dropdown-item" id="btn-open-cadastrar-usuario" href="#">Cadastrar</a>
                        <li><a class="dropdown-item" id="listar-usuarios-link" href="#">Listar Usuários</a>
                    </ul>
                </li>
            </ul>
        </div>
    </div>
</nav>

```

```

        </ul>
    </li>
    <!-- NOVO: Setores -->
    <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="setoresDropdown" data-bs-toggle="dr
        <ul class="dropdown-menu" aria-labelledby="setoresDropdown">
            <li><a class="dropdown-item" id="btn-open-cadastrar-setor" href="#">Cadastrar Se
            <li><a class="dropdown-item" id="listar-setores-link" href="#">Listar Setores</a>
        </ul>
    </li>
    <!-- Itens (apenas leitura) -->
    <li class="nav-item">
        <a class="nav-link" href="#" id="listar-item-link">LISTAR ITENS</a>
    </li>
    <!-- Retiradas -->
    <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="retiradasDropdown" data-bs-toggle="
        <ul class="dropdown-menu" aria-labelledby="retiradasDropdown">
            <li><a class="dropdown-item" id="btn-open-solicitar-retirada" href="#">Solicitar
            <li><a class="dropdown-item" id="listar-retiradas-link" href="#">Histórico de Re
        </ul>
    </li>
    <!-- Relatórios -->
    <li class="nav-item">
        <a class="nav-link" href="#" id="open-reports-dashboard"
            data-bs-toggle="modal" data-bs-target="#modalReportsDashboard">RELATÓRIOS</a>
    </li>
    <!-- Alertas -->
    <li class="nav-item">
        <a class="nav-link" href="#" id="open-alertas-modal">HISTÓRICO DE ALERTAS</a>
    </li>
</ul>
<div class="d-flex align-items-center">
    <div class="dropdown">
        <a class="nav-link position-relative me-3" href="#" id="alert-notification-bell" rol
            <i class="bi bi-bell-fill fs-5"></i>
            <span id="notification-dot" class="notification-dot position-absolute">
                <span class="visually-hidden">Novas notificações</span>
            </span>
        </a>
        <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="alert-notification-bell
            <li><span class="dropdown-item text-muted" id="no-notifications-menu-item" style
            <li><a class="dropdown-item" href="#" id="new-alerts-menu-item" style="display:
            <li><hr class="dropdown-divider"></li>
            <!-- Adicione aqui o link para "Novas Solicitações de Retirada" se necessário pa
            <li><a class="dropdown-item" href="#" id="new-withdrawal-requests-menu-item" sty
            <li><hr class="dropdown-divider"></li>
        </ul>
    </div>
    <div class="dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="userMenu" data-bs-toggle="dropdown"
            aria-expanded="false">
            <i class="bi bi-person-circle"></i>
        </a>
        <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="userMenu">
            <li><a class="dropdown-item" href="#" id="logout-link">Sair</a></li>
        </ul>
    </div>
</div>
</div>
</div>
</nav>

```


Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\navbar_servidor.html

```
<nav class="navbar navbar-expand-lg navbar-dark navbar-custom">
  <div class="container-fluid">
    <button class="btn btn-outline-light me-3" id="home-button"><i class="bi bi-house-fill"></i>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
      aria-controls="navbarMenu" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarMenu">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">

        <li class="nav-item">
          <a class="nav-link" href="#" id="solicitar-retirada-servidor-link" >
            SOLICITAR RETIRADA DE ITENS
          </a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#" id="historico-retiradas-servidor-link" >
            MEU HISTÓRICO DE RETIRADAS
          </a>
        </li>

      </ul>
      <div class="d-flex align-items-center">
        <div class="dropdown">
          <a class="nav-link position-relative me-3" href="#" id="alert-notification-bell">
            <i class="bi bi-bell-fill fs-5"></i>
          </a>
          <span id="notification-dot" class="notification-dot position-absolute">
            <span class="visually-hidden">Novas notificações</span>
          </span>
          <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="alert-notification-bell">
            <li><span class="dropdown-item text-muted" id="no-notifications-menu-item">Sem notificações</span></li>
            <li><a class="dropdown-item" href="#" id="new-withdrawal-requests-menu-item">Novas solicitações de retirada</a></li>
            <li><a class="dropdown-item" href="#" id="new-alerts-menu-item" style="display: none;">Novas alertas</a></li>
            <li><hr class="dropdown-divider"></li>
          </ul>
        </div>
        <div class="dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="userMenu" data-bs-toggle="dropdown">
            <i class="bi bi-person-circle"></i>
          </a>
          <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="userMenu">
            <li><a class="dropdown-item" href="#" id="logout-link">Sair</a></li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</nav>
```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\nav_items.html

```
<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" id="itensDropdown" data-bs-toggle="dropdown">ITENS</a>
  <ul class="dropdown-menu" aria-labelledby="itensDropdown">
```

```

        <li><a class="dropdown-item" id="btn-open-cadastrar-item" href="#">Cadastrar Item</a></li>
        <li><a class="dropdown-item" id="listar-item-link" href="#">Listar Itens</a></li>
        <li><hr class="dropdown-divider"></li>
        <li><a class="dropdown-item" id="btn-open-importar-tabela" href="#">Importar Tabela</a></li>
    </ul>
</li>

<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" href="#" id="categoriasDropdown" data-bs-toggle="dropdown">CATEGORIAS</a>
    <ul class="dropdown-menu" aria-labelledby="categoriasDropdown">
        <li><a class="dropdown-item" id="btn-open-cadastrar-categoria" href="#">Cadastrar Categoria</a></li>
        <li><a class="dropdown-item" id="listar-categoria-link-quick" href="#">Listar Categoria</a></li>
    </ul>
</li>

<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" href="#" id="retiradasDropdown" data-bs-toggle="dropdown">RETIRADAS</a>
    <ul class="dropdown-menu" aria-labelledby="retiradasDropdown">
        <li><a class="dropdown-item" id="btn-open-solicitar-retirada" href="#">Solicitar Retirada</a></li>
        <li><a class="dropdown-item" id="listar-retiradas-link" href="#">Histórico de Retiradas</a></li>
        <li><a class="dropdown-item" id="listar-retiradas-pendentes-link" href="#">Solicitações Pendentes</a></li>
    </ul>
</li>

<li class="nav-item">
    <a class="nav-link" href="#" id="open-reports-dashboard" data-bs-toggle="modal" data-bs-target="#modalReportsDashboard" >RELATÓRIOS</a>
</li>

<li class="nav-item dropdown">
    <a class="nav-link" href="#" id="open-alertas-modal" >HISTÓRICO DE ALERTAS</a>
</li>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\quick_access.html

```

<!-- frontend\templates\partials\quick_access.html -->
<section class="container py-4" id="main-content">
    <h3 class="mb-3">Acesso Rápido</h3>
    <div class="row g-3">
        {% include 'partials/quick_cards.html' %}
    </div>
</section>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\quick_access_direcao.html

```

<!-- frontend\templates\partials\quick_access.html -->
<section class="container py-4" id="main-content">
    <h3 class="mb-3">Acesso Rápido</h3>
    <div class="row g-3">
        {% include 'partials/quick_cards_direcao.html' %}
    </div>
</section>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\quick_cards.html

```
<!-- frontend\templates\partials\quick_cards.html -->
<div class="col-6 col-md-3">
  <a href="#" id="btn-open-cadastrar-item" class="text-decoration-none">
    <div class="quick-access-card">
      <i class="bi bi-plus-square"></i>
      <p>Cadastrar Item</p>
    </div>
  </a>
</div>
<div class="col-6 col-md-3">
  <a href="#" id="listar-item-link-quick" class="text-decoration-none">
    <div class="quick-access-card">
      <i class="bi bi-card-list"></i>
      <p>Listar Itens</p>
    </div>
  </a>
</div>
<div class="col-6 col-md-3">
  <a href="#" id="listar-retiradas-pendentes-quick" class="text-decoration-none">
    <div class="quick-access-card">
      <i class="bi bi-card-checklist"></i>
      <p>Retiradas Pendentes</p>
    </div>
  </a>
</div>
<div class="col-6 col-md-3">
  <a href="#" id="open-reports-dashboard" data-bs-toggle="modal" data-bs-target="#modalReportsDashboard">
    <div class="quick-access-card">
      <i class="bi bi-bar-chart-line"></i>
      <p>Relatórios</p>
    </div>
  </a>
</div>
```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\quick_cards_direcao.html

```
<!-- frontend\templates\partials\quick_cards.html -->
<div class="col-6 col-md-3">
  <a href="#" id="listar-usuarios-link" class="text-decoration-none">
    <div class="quick-access-card">
      <i class="bi bi-person-fill-gear"></i>
      <p>Gerenciar Usuários</p>
    </div>
  </a>
</div>
<div class="col-6 col-md-3">
  <a href="#" id="listar-setores-link" class="text-decoration-none">
    <div class="quick-access-card">
      <i class="bi bi-building-fill"></i>
      <p>Gerenciar Setores</p>
    </div>
  </a>
</div>
<div class="col-6 col-md-3">
  <a href="#" id="open-alertas-modal" class="text-decoration-none">
```

```

        <div class="quick-access-card">
            <i class="bi bi-exclamation-octagon-fill"></i>
            <p>Alertas de Estoque</p>
        </div>
    </a>
</div>
<div class="col-6 col-md-3">
    <a href="#" id="open-reports-dashboard" data-bs-toggle="modal" data-bs-target="#modalReportsDashboard">
        <div class="quick-access-card">
            <i class="bi bi-bar-chart-line-fill"></i>
            <p>Relatórios</p>
        </div>
    </a>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\scripts.html

```

<script src="/static/js/bootstrap.bundle.min.js"></script>
<script src="/static/js/logout.js"></script>
<script src="/static/js/listar-itens.js"></script>
<script src="/static/js/listar-categorias.js"></script>
<script src="/static/js/cadastrar-item.js"></script>
<script src="/static/js/cadastrar-categorias.js"></script>

<script type="module" src="/static/js/estadoGlobal.js"></script>
<script type="module" src="/static/js/utils.js"></script>
<script type="module" src="/static/js/apiService.js"></script>
<script type="module" src="/static/js/dataService.js"></script>
<script type="module" src="/static/js/uiService.js"></script>
<script type="module" src="/static/js/retiradasModule.js"></script>
<script type="module" src="/static/js/solicitar-retirada.js"></script>
<script type="module" src="/static/js/selecionar-item-module.js"></script>
<script type="module" src="/static/js/reportsModule.js"></script>
<script type="module" src="/static/js/main.js"></script>

</body>
</html>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\topbar.html

```

<!-- frontend\templates\partials\topbar.html -->
<header class="top-bar d-flex justify-content-between align-items-center">
    <div class="logo">Sistema de Controle de Almoxarifado</div>
    <div class="d-flex align-items-center">
        <span class="me-2">Centro Profissional e Tecnológico</span>
        
    </div>
</header>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_autorizar_retirada.html

```

<div class="modal fade" id="modalAutorizarRetirada" tabindex="-1" aria-labelledby="modalAutorizarRetiradaLabel">
  <div class="modal-dialog modal-lg modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalAutorizarRetiradaLabel">Autorizar Retirada</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
      </div>
      <div class="modal-body">
        <form class="row g-3">
          <input type="hidden" id="autorizarRetiradaId">

          <div class="col-md-6">
            <label for="autorizarSetor" class="form-label">Setor da Solicitação:</label>
            <input type="text" class="form-control" id="autorizarSetor" readonly>
          </div>
          <div class="col-md-6">
            <label for="autorizarUsuario" class="form-label">Solicitado por:</label>
            <input type="text" class="form-control" id="autorizarUsuario" readonly>
          </div>
          <div class="col-12">
            <label for="autorizarJustificativa" class="form-label">Justificativa da Solicitação</label>
            <textarea class="form-control" id="autorizarJustificativa" rows="2" readonly></textarea>
          </div>
          <div class="col-12">
            <label for="autorizarData" class="form-label">Data da Solicitação</label>
            <input type="text" class="form-control" id="autorizarData" readonly>
          </div>
          <div class="col-12">
            <label class="form-label">Itens Solicitados</label>
            <div id="autorizarItens" class="list-group"></div>
          </div>
          <div class="col-12">
            <label for="autorizarDetalheStatus" class="form-label">Justificativa para Negar/Autorizar</label>
            <textarea class="form-control" id="autorizarDetalheStatus" rows="2"></textarea>
          </div>
        </form>
      </div>
      <div class="modal-footer border-0 justify-content-center">
        <button id="btn-confirmar-negar-retirada" class="btn btn-danger me-2">Negar</button>
        <button id="btn-confirmar-autorizar-retirada" class="btn btn-success me-2">Autorizar</button>
        <button class="btn btn-secondary" data-bs-dismiss="modal">Fechar</button>
      </div>
    </div>
  </div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_cadastrar_categoria.html

```

<div class="modal fade" id="modalCadastrarCategoria" tabindex="-1" aria-labelledby="modalCadastrarCategoriaLabel">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalCadastrarCategoriaLabel">Cadastrar Categoria</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal"></button>
      </div>
      <div class="modal-body">
        <form id="form-cadastrar-categoria">
          <div class="mb-3">

```

```

        <label for="nome_categoria" class="form-label">Nome*</label>
        <input type="text" class="form-control" id="nome_categoria" name="nome_categoria" required>
    </div>
    <div class="mb-3">
        <label for="descricao_categoria" class="form-label">Descrição</label>
        <textarea class="form-control" id="descricao_categoria" name="descricao_categoria" rows="2">
    </div>
    </form>
</div>
<div class="modal-footer border-0 justify-content-center">
    <button class="btn btn-secondary me-2" id="btn-cancelar_categoria">Cancelar</button>
    <button class="btn btn-primary" id="btn-salvar_categoria">Salvar</button>
</div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_cadastrar_item.html

```

<div class="modal fade" id="modalCadastrarItem" tabindex="-1" aria-labelledby="modalCadastrarItemLabel">
    <div class="modal-dialog modal-lg modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <div class="modal-header border-0">
                <h5 class="modal-title" id="modalCadastrarItemLabel">Cadastrar Novo Item</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
            </div>
            <div class="modal-body">
                <form id="form-cadastrar-item">
                    <div class="row g-3">
                        <div class="col-md-6">
                            <label for="nome_item" class="form-label">Nome do Item*</label>
                            <input type="text" class="form-control" id="nome_item" name="nome_item" required>
                        </div>
                        <div class="col-md-6">
                            <label for="unidade_medida_item" class="form-label">Unidade de Medida*</label>
                            <input type="text" class="form-control" id="unidade_medida_item" name="unidade_medida_item">
                        </div>
                        <div class="col-md-6">
                            <label for="descricao_item" class="form-label">Descrição*</label>
                            <textarea class="form-control" id="descricao_item" name="descricao_item" rows="2" required>
                        </div>
                        <div class="col-md-3">
                            <label for="quantidade_item" class="form-label">Quantidade*</label>
                            <input type="number" class="form-control" id="quantidade_item" name="quantidade_item" min="1">
                        </div>
                        <div class="col-md-3">
                            <label for="quantidade_minima_item" class="form-label">Quantidade Mínima</label>
                            <input type="number" class="form-control" id="quantidade_minima_item" name="quantidade_minima_item">
                        </div>
                        <div class="col-md-4">
                            <label for="data_validade_item" class="form-label">Data de Validade</label>
                            <input type="date" class="form-control" id="data_validade_item" name="data_validade_item">
                        </div>
                        <div class="col-md-4">
                            <label for="data_entrada_item" class="form-label">Data de Entrada</label>
                            <input type="datetime-local" class="form-control" id="data_entrada_item" name="data_entrada_item">
                        </div>
                        <div class="col-md-4">

```

```

        <label for="marca_item" class="form-label">Marca</label>
        <input type="text" class="form-control" id="marca_item" name="marca_item">
    </div>
    <div class="col-md-6">
        <label for="categoria_id" class="form-label">Categoria*</label>
        <select class="form-select" id="categoria_id" name="categoria_id" required>
            <option value="" disabled selected></option>
        </select>
    </div>
</div>
</form>
</div>
<div class="modal-footer border-0 justify-content-center">
    <button type="button" class="btn btn-secondary me-2" id="btn-cancelar-item">Cancelar</button>
    <button type="button" class="btn btn-primary" id="btn-salvar-item">Enviar</button>
</div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_cadastrar_setor.html

```

<div class="modal fade" id="modalCadastrarSetor" tabindex="-1" aria-labelledby="modalCadastrarSetorLabel">
    <div class="modal-dialog modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <div class="modal-header border-0">
                <h5 class="modal-title" id="modalCadastrarSetorLabel">Cadastrar Novo Setor</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
            </div>
            <div class="modal-body">
                <form id="form-cadastrar-setor">
                    <div class="mb-3">
                        <label for="cad_nome_setor" class="form-label">Nome do Setor*</label>
                        <input type="text" class="form-control" id="cad_nome_setor" name="nome_setor" required="">
                    </div>
                    <div class="mb-3">
                        <label for="cad_descricao_setor" class="form-label">Descrição (Opcional)</label>
                        <textarea class="form-control" id="cad_descricao_setor" name="descricao_setor" rows="3">
                    </div>
                </form>
            </div>
            <div class="modal-footer border-0 justify-content-center">
                <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
                <button type="button" class="btn btn-primary" id="btn-salvar-cadastrar-setor">Salvar</button>
            </div>
        </div>
    </div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_cadastrar_usuario.html

```

<!-- frontend/templates/partials/modals/modal_cadastrar_usuario.html -->

```

```

<div class="modal fade" id="modalCadastrarUsuario" tabindex="-1" aria-labelledby="modalCadastrarUsuarioLabel">
  <div class="modal-dialog modal-lg modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalCadastrarUsuarioLabel">Cadastrar Novo Usuário</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
      </div>
      <div class="modal-body">
        <form id="form-cadastrar-usuario" class="row g-3">
          <div class="col-md-6">
            <label for="cad_nome_usuario" class="form-label">Nome Completo*</label>
            <input type="text" class="form-control" id="cad_nome_usuario" name="nome_usuario">
          </div>
          <div class="col-md-6">
            <label for="cad_email_usuario" class="form-label">Email*</label>
            <input type="email" class="form-control" id="cad_email_usuario" name="email_usuario">
          </div>
          <div class="col-md-6">
            <label for="cad_username" class="form-label">Username*</label>
            <input type="text" class="form-control" id="cad_username" name="username" required="">
          </div>
          <div class="col-md-6">
            <label for="cad_senha_usuario" class="form-label">Senha*</label>
            <input type="password" class="form-control" id="cad_senha_usuario" name="senha_usuario">
          </div>
          <div class="col-md-6">
            <label for="cad_tipo_usuario" class="form-label">Tipo de Usuário*</label>
            <select class="form-select" id="cad_tipo_usuario" name="tipo_usuario" required="">
              <option value="" disabled selected>Selecione o tipo</option>
              <option value="1">Usuário Geral</option>
              <option value="2">Almoxarifado</option>
              <option value="3">Direção</option>
            </select>
          </div>
          <div class="col-md-6">
            <label for="cad_setor_id" class="form-label">Setor*</label>
            <select class="form-select" id="cad_setor_id" name="setor_id" required="">
              <option value="" disabled selected>Selecione um setor</option>
            </select>
          </div>
          <div class="col-md-6">
            <label for="cad_siape_usuario" class="form-label">SIAPE (Opcional)</label>
            <input type="number" class="form-control" id="cad_siape_usuario" name="siape_usuario">
          </div>
        </form>
      </div>
      <div class="modal-footer border-0 justify-content-center">
        <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
        <button type="button" class="btn btn-primary" id="btn-salvar-cadastrar-usuario">Salvar</button>
      </div>
    </div>
  </div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_concluir_retirada.html

```

<div class="modal fade" id="modalConcluirRetirada" tabindex="-1" aria-labelledby="modalConcluirRetiradaLabel">
  <div class="modal-dialog modal-dialog-centered">

```



```

<div class="modal-content rounded-4 shadow">
  <div class="modal-header border-0">
    <h5 class="modal-title" id="modalConcluirRetiradaLabel">Concluir Retirada</h5>
    <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
  </div>
  <div class="modal-body">
    <form id="form-concluir-retirada">
      <input type="hidden" id="concluirRetiradaId">
      <p>Você tem certeza que deseja concluir a retirada ID: <strong id="concluirRetiradaID"></strong></p>
      <p>Esta ação decreta o estoque dos itens!</p>
      <div class="mb-3">
        <label for="concluirDetalheStatus" class="form-label">Detalhe do Status (Opcional)</label>
        <textarea class="form-control" id="concluirDetalheStatus" rows="3" placeholder="Detalhe do Status (Opcional)"></textarea>
      </div>
    </form>
  </div>
  <div class="modal-footer border-0 justify-content-center">
    <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
    <button type="button" class="btn btn-success" id="btn-confirmar-concluir-retirada">Confirmar</button>
  </div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_confirmar_deletar_setor.html

```

<div class="modal fade" id="modalConfirmarDeleteSetor" tabindex="-1" aria-labelledby="modalConfirmarDeleteSetorLabel">
  <div class="modal-dialog modal-sm modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalConfirmarDeleteSetorLabel">Confirmar Exclusão</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
      </div>
      <div class="modal-body text-center">
        <p>Tem certeza que deseja excluir o setor com ID: <strong id="confirm-delete-setor-id"></strong></p>
        <p class="text-danger">Esta ação é irreversível!</p>
      </div>
      <div class="modal-footer border-0 justify-content-center">
        <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
        <button type="button" class="btn btn-danger" id="btn-confirmar-deletar-setor">Excluir</button>
      </div>
    </div>
  </div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_confirmar_deletar_usuario.html

```

<!-- frontend/templates/partials/modals/modal_confirmar_deletar_usuario.html -->
<div class="modal fade" id="modalConfirmarDeleteUsuario" tabindex="-1" aria-labelledby="modalConfirmarDeleteUsuarioLabel">
  <div class="modal-dialog modal-sm modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">

```

```

        <h5 class="modal-title" id="modalConfirmarDeleteUsuarioLabel">Confirmar Exclusão</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
    </div>
    <div class="modal-body text-center">
        <p>Tem certeza que deseja excluir o usuário com ID: <strong id="confirm-delete-user-id"></strong></p>
        <p class="text-danger">Esta ação é irreversível!</p>
    </div>
    <div class="modal-footer border-0 justify-content-center">
        <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
        <button type="button" class="btn btn-danger" id="btn-confirmar-deletar-usuario">Excluir</button>
    </div>
</div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_editar_categoria.html

```

<div class="modal fade" id="modalEditarCategoria" tabindex="-1" aria-labelledby="modalEditarCategoriaLabel">
    <div class="modal-dialog modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <div class="modal-header border-0">
                <h5 class="modal-title" id="modalEditarCategoriaLabel">Editar Categoria</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal"></button>
            </div>
            <div class="modal-body">
                <form id="form-editar-categoria">
                    <div class="mb-3">
                        <label for="edit-nome_categoria" class="form-label">Nome*</label>
                        <input type="text" class="form-control" id="edit-nome_categoria" name="nome_categoria" required/>
                    </div>
                    <div class="mb-3">
                        <label for="edit-descricao_categoria" class="form-label">Descrição</label>
                        <textarea class="form-control" id="edit-descricao_categoria" name="descricao_categoria" rows="3"></textarea>
                    </div>
                </form>
            </div>
            <div class="modal-footer border-0 justify-content-center">
                <button class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
                <button class="btn btn-primary" id="btn-salvar-editar-categoria">Salvar</button>
            </div>
        </div>
    </div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_editar_item.html

```

<!-- frontend/templates/partials/modals/modal_editar_item.html -->

<div class="modal fade" id="modalEditarItem" tabindex="-1" aria-labelledby="modalEditarItemLabel" aria-hidden="true">
    <div class="modal-dialog modal-lg modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <!-- Modal Header -->

```

```

<div class="modal-header border-0">
  <h5 class="modal-title" id="modalEditarItemLabel">Editar Item</h5>
  <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
</div>

<!-- Modal Body -->
<div class="modal-body">
  <form id="form-editar-item">
    <div class="row g-3">
      <!-- Campos de edição -->
      <div class="col-md-6">
        <label for="nome_item" class="form-label">Nome do Item*</label>
        <input type="text" class="form-control" id="nome_item" name="nome_item" required>
      </div>
      <div class="col-md-6">
        <label for="unidade_medida_item" class="form-label">Unidade de Medida*</label>
        <input type="text" class="form-control" id="unidade_medida_item" name="unidade_medida_item">
      </div>
      <div class="col-md-6">
        <label for="descricao_item" class="form-label">Descrição*</label>
        <textarea class="form-control" id="descricao_item" name="descricao_item" rows="2" required>
      </div>
      <div class="col-md-3">
        <label for="quantidade_item" class="form-label">Quantidade*</label>
        <input type="number" class="form-control" id="quantidade_item" name="quantidade_item" min="1">
      </div>
      <div class="col-md-3">
        <label for="quantidade_minima_item" class="form-label">Quantidade Mínima</label>
        <input type="number" class="form-control" id="quantidade_minima_item" name="quantidade_minima_item">
      </div>
      <div class="col-md-4">
        <label for="data_validade_item" class="form-label">Data de Validade</label>
        <input type="date" class="form-control" id="data_validade_item" name="data_validade_item">
      </div>
      <div class="col-md-4">
        <label for="data_entrada_item" class="form-label">Data de Entrada</label>
        <input type="datetime-local" class="form-control" id="data_entrada_item" name="data_entrada_item">
      </div>
      <div class="col-md-4">
        <label for="marca_item" class="form-label">Marca</label>
        <input type="text" class="form-control" id="marca_item" name="marca_item">
      </div>
      <div class="col-md-6">
        <label for="edit-categoria_id" class="form-label">Categoria*</label>
        <select class="form-select" id="edit-categoria_id" name="categoria_id" required>
          <option value="" disabled selected></option>
        </select>
      </div>
    </div>
  </form>
</div>

<!-- Modal Footer -->
<div class="modal-footer border-0 justify-content-center">
  <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
  <button type="button" class="btn btn-primary" id="btn-salvar-editar-item">Salvar</button>
</div>

</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_editar_setor.html

```
<div class="modal fade" id="modalEditarSetor" tabindex="-1" aria-labelledby="modalEditarSetorLabel" aria-
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalEditarSetorLabel">Editar Setor</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
      </div>
      <div class="modal-body">
        <form id="form-editar-setor">
          <div class="mb-3">
            <label for="edit_nome_setor" class="form-label">Nome do Setor*</label>
            <input type="text" class="form-control" id="edit_nome_setor" name="nome_setor" r
          </div>
          <div class="mb-3">
            <label for="edit_descricao_setor" class="form-label">Descrição (Opcional)</label>
            <textarea class="form-control" id="edit_descricao_setor" name="descricao_setor"
          </div>
        </form>
      </div>
      <div class="modal-footer border-0 justify-content-center">
        <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
        <button type="button" class="btn btn-primary" id="btn-salvar-editar-setor">Salvar Altera
      </div>
    </div>
  </div>
</div>
```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_editar_usuario.html

```
<!-- frontend/templates/partials/modals/modal_editar_usuario.html -->
<div class="modal fade" id="modalEditarUsuario" tabindex="-1" aria-labelledby="modalEditarUsuarioLabel"
  <div class="modal-dialog modal-lg modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalEditarUsuarioLabel">Editar Usuário</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
      </div>
      <div class="modal-body">
        <form id="form-editar-usuario" class="row g-3">
          <div class="col-md-6">
            <label for="edit_nome_usuario" class="form-label">Nome Completo*</label>
            <input type="text" class="form-control" id="edit_nome_usuario" name="nome_usuario"
          </div>
          <div class="col-md-6">
            <label for="edit_email_usuario" class="form-label">Email*</label>
            <input type="email" class="form-control" id="edit_email_usuario" name="email_usuario"
          </div>
          <div class="col-md-6">
            <label for="edit_username" class="form-label">Username*</label>
            <input type="text" class="form-control" id="edit_username" name="username" requi
          </div>
          <div class="col-md-6">

```

```

        <label for="edit_senha_usuario" class="form-label">Nova Senha (Opcional)</label>
        <input type="password" class="form-control" id="edit_senha_usuario" name="senha_usuario">
        <small class="form-text text-muted">Deixe em branco para manter a senha atual.</small>
    </div>
    <div class="col-md-6">
        <label for="edit_tipo_usuario" class="form-label">Tipo de Usuário</label>
        <select class="form-select" id="edit_tipo_usuario" name="tipo_usuario" required>
            <option value="1">Usuário Geral</option>
            <option value="2">Almoxarifado</option>
            <option value="3">Direção</option>
        </select>
    </div>
    <div class="col-md-6">
        <label for="edit_setor_id" class="form-label">Setor</label>
        <select class="form-select" id="edit_setor_id" name="setor_id" required>
            <option value="" disabled selected>Selecione um setor</option>
        </select>
    </div>
    <div class="col-md-6">
        <label for="edit_siape_usuario" class="form-label">SIAPE (Opcional)</label>
        <input type="number" class="form-control" id="edit_siape_usuario" name="siape_usuario">
    </div>
</form>
</div>
<div class="modal-footer border-0 justify-content-center">
    <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
    <button type="button" class="btn btn-primary" id="btn-salvar-editar-usuario">Salvar Alterações</button>
</div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_item_detalhes.html

```

<!-- frontend/templates/partials/modals/modal_item_detalhes.html -->
<div class="modal fade" id="modalDetalheItem" tabindex="-1" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <div class="modal-header border-0">
                <h5 class="modal-title">Detalhes do Item</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal"></button>
            </div>
            <div class="modal-body">
                <ul class="list-unstyled mb-0">
                    <li><strong>Nome:</strong> <span id="itemNome"></span></li>

                    <!-- escondido por padrão -->
                    <li id="liItemEstoque" style="display: none;">
                        <strong>Em Estoque:</strong> <span id="itemEstoque"></span>
                    </li>

                    <li><strong>Solicitado:</strong> <span id="itemQtdRetirada"></span></li>

                    <!-- escondido por padrão -->
                    <li id="liItemEstoqueMin" style="display: none;">
                        <strong>Estoque Mínimo:</strong> <span id="itemEstoqueMin"></span>
                    </li>

                    <li><strong>Validade:</strong> <span id="itemValidade"></span></li>
                </ul>
            </div>
        </div>
    </div>
</div>

```

```

        </ul>
    </div>
    <div class="modal-footer border-0">
        <button class="btn btn-secondary" data-bs-dismiss="modal">Fechar</button>
    </div>
</div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_reports_dashboard.html

```

<div class="modal fade" id="modalReportsDashboard" tabindex="-1" aria-labelledby="modalReportsDashboardLabel">
    <div class="modal-dialog modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <div class="modal-header border-0">
                <h5 class="modal-title" id="modalReportsDashboardLabel">Gerar Relatórios</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
            </div>
            <div class="modal-body">
                <div class="list-group list-group-flush"> {# Adicionado list-group-flush para remover border<
                    <button type="button" class="list-group-item list-group-item-action py-3 px-4 d-flex align-items-between">
                        Relatório de Quantidade de Itens
                        <i class="bi bi-box-seam text-primary fs-4"></i> {# Ícone de sua escolha #}
                    </button>
                    <button type="button" class="list-group-item list-group-item-action py-3 px-4 d-flex align-items-between">
                        Relatório de Entrada de Itens
                        <i class="bi bi-arrow-down-square text-success fs-4"></i> {# Ícone de sua escolha #}
                    </button>
                    <button type="button" class="list-group-item list-group-item-action py-3 px-4 d-flex align-items-between">
                        Relatório de Retiradas por Setor
                        <i class="bi bi-building text-info fs-4"></i> {# Ícone de sua escolha #}
                    </button>
                    <button type="button" class="list-group-item list-group-item-action py-3 px-4 d-flex align-items-between">
                        Relatório de Retiradas por Usuário
                        <i class="bi bi-person-fill text-warning fs-4"></i> {# Ícone de sua escolha #}
                    </button>
                </div>
            </div>
            <div class="modal-footer border-0 justify-content-center">
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Fechar</button>
            </div>
        </div>
    </div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_report_entrada_itens.html

```

<div class="modal fade" id="modalReportEntradaItens" tabindex="-1" aria-labelledby="modalReportEntradaItensLabel">
    <div class="modal-dialog modal-md modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <div class="modal-header border-0">
                <h5 class="modal-title" id="modalReportEntradaItensLabel">Relatório de Entrada de Itens</h5>
            </div>

```

```

        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
    </div>
    <div class="modal-body">
        <form id="form-report-entrada-itens">
            <div class="mb-3">
                <label for="dataInicioEntrada" class="form-label">Data Inicial*</label>
                <input type="date" class="form-control" id="dataInicioEntrada" name="data_inicio">
            </div>
            <div class="mb-3">
                <label for="dataFimEntrada" class="form-label">Data Final*</label>
                <input type="date" class="form-control" id="dataFimEntrada" name="data_fim" required>
            </div>
            <div class="mb-3">
                <label for="formatoEntradaItens" class="form-label">Formato do Relatório*</label>
                <select class="form-select" id="formatoEntradaItens" name="formato" required>
                    <option value="csv">CSV</option>
                    <option value="xlsx" selected>XLSX</option>
                </select>
            </div>
        </form>
    </div>
    <div class="modal-footer border-0 justify-content-center">
        <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
        <button type="button" class="btn btn-primary" id="btnGerarReportEntradaItens">Gerar Relatório</button>
    </div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_report_quantidade_itens.html

```

<div class="modal fade" id="modalReportQuantidadeItens" tabindex="-1" aria-labelledby="modalReportQuantidadeItensLabel">
    <div class="modal-dialog modal-md modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <div class="modal-header border-0">
                <h5 class="modal-title" id="modalReportQuantidadeItensLabel">Relatório de Quantidade de Itens</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
            </div>
            <div class="modal-body">
                <form id="form-report-quantidade-itens">
                    <div class="mb-3">
                        <label for="filterCategoryReport" class="form-label">Filtrar por Categoria (opcional)</label>
                        <select class="form-select" id="filterCategoryReport" name="filtro_categoria">
                            <option value="">Todas as Categorias</option>
                            <option value="1">Categoria 1</option>
                            <option value="2">Categoria 2</option>
                        </select>
                    </div>
                    <div class="mb-3">
                        <label for="selectedProductNameReport" class="form-label">Filtrar por Produto (opcional)</label>
                        <div class="input-group">
                            <input type="text" class="form-control" id="selectedProductNameReport" name="filtro_produto">
                            <button class="btn btn-outline-secondary" type="button" id="btnOpenSelectProduct">
                                <i class="bi bi-search"></i> Selecionar
                            </button>
                            <button class="btn btn-outline-danger" type="button" id="btnClearSelectedProduct">
                                <i class="bi bi-x-lg"></i> Limpar
                            </button>
                        </div>
                    </div>
                    <input type="hidden" id="selectedProductIdReport" name="filtro_produto_id">
                </form>
            </div>
        </div>
    </div>
</div>

```

```

        </div>
        <div class="mb-3">
            <label for="formatoQuantidadeItens" class="form-label">Formato do Relatório*</label>
            <select class="form-select" id="formatoQuantidadeItens" name="formato" required>
                <option value="csv">CSV</option>
                <option value="xlsx" selected>XLSX</option>
            </select>
        </div>
    </form>
</div>
<div class="modal-footer border-0 justify-content-center">
    <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
    <button type="button" class="btn btn-primary" id="btnGerarReportQuantidadeItens">Gerar R
</div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_report_retiradas_setor.html

```

<div class="modal fade" id="modalReportRetiradasSetor" tabindex="-1" aria-labelledby="modalReportRetiradasSetorLabel">
    <div class="modal-dialog modal-md modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <div class="modal-header border-0">
                <h5 class="modal-title" id="modalReportRetiradasSetorLabel">Relatório de Retiradas por Setor</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
            </div>
            <div class="modal-body">
                <form id="form-report-retiradas-setor">
                    <div class="mb-3">
                        <label for="selectSetor" class="form-label">Setor*</label>
                        <select class="form-select" id="selectSetor" name="setor_id" required>
                            <option value="" disabled selected>Carregando setores...</option>
                        </select>
                    </div>
                    <div class="mb-3">
                        <label for="dataInicioRetiradaSetor" class="form-label">Data Inicial*</label>
                        <input type="date" class="form-control" id="dataInicioRetiradaSetor" name="data_inicio">
                    </div>
                    <div class="mb-3">
                        <label for="dataFimRetiradaSetor" class="form-label">Data Final*</label>
                        <input type="date" class="form-control" id="dataFimRetiradaSetor" name="data_fim">
                    </div>
                    <div class="mb-3">
                        <label for="formatoRetiradasSetor" class="form-label">Formato do Relatório*</label>
                        <select class="form-select" id="formatoRetiradasSetor" name="formato" required>
                            <option value="csv">CSV</option>
                            <option value="xlsx" selected>XLSX</option>
                        </select>
                    </div>
                </form>
            </div>
            <div class="modal-footer border-0 justify-content-center">
                <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
                <button type="button" class="btn btn-primary" id="btnGerarReportRetiradasSetor">Gerar Relatório</button>
            </div>
        </div>
    </div>
</div>

```


</div>

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_report_retiradas_usuario.html

```
<div class="modal fade" id="modalReportRetiradasUsuario" tabindex="-1" aria-labelledby="modalReportRetiradasUsuarioLabel">
  <div class="modal-dialog modal-md modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalReportRetiradasUsuarioLabel">Relatório de Retiradas por Usuário*
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
      </div>
      <div class="modal-body">
        <form id="form-report-retiradas-usuario">
          <div class="mb-3">
            <label for="selectUsuario" class="form-label">Usuário*
            <select class="form-select" id="selectUsuario" name="usuario_id" required>
              <option value="" disabled selected>Carregando usuários...
            </select>
          </div>
          <div class="mb-3">
            <label for="dataInicioRetiradaUsuario" class="form-label">Data Inicial*
            <input type="date" class="form-control" id="dataInicioRetiradaUsuario" name="data_inicio">
          </div>
          <div class="mb-3">
            <label for="dataFimRetiradaUsuario" class="form-label">Data Final*
            <input type="date" class="form-control" id="dataFimRetiradaUsuario" name="data_fim">
          </div>
          <div class="mb-3">
            <label for="formatoRetiradasUsuario" class="form-label">Formato do Relatório*
            <select class="form-select" id="formatoRetiradasUsuario" name="formato" required>
              <option value="csv">CSV
              <option value="xlsx" selected>XLSX
            </select>
          </div>
        </form>
      </div>
      <div class="modal-footer border-0 justify-content-center">
        <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar
        <button type="button" class="btn btn-primary" id="btnGerarReportRetiradasUsuario">Gerar Relatório
      </div>
    </div>
  </div>
</div>
```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_selecionar_item_retirada.html

```
<!-- frontend\templates\partials\modals\modal_selecionar_item_retirada.html -->
<div class="modal fade" id="modalSelecionarItemRetirada" tabindex="-1" aria-labelledby="modalSelecionarItemRetiradaLabel">
  <div class="modal-dialog modal-md modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalSelecionarItemRetiradaLabel">Selecionar Item para Retirada
      </div>
    </div>
  </div>
</div>
```

```

        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
    </div>
    <div class="modal-body">
        <form id="form-search-item-retirada" class="mb-3">
            <div class="row g-3">
                <div class="col-md-7">
                    <label for="searchItemName" class="form-label visually-hidden">Buscar por Nome</label>
                    <input type="text" class="form-control" id="searchItemName" placeholder="Busque aqui o nome do item">
                </div>
                <div class="col-md-5 d-flex">
                    <button type="submit" class="btn btn-primary flex-grow-1 me-2" id="btn-search-item">Pesquisar</button>
                    <button type="button" class="btn btn-secondary" id="btn-clear-item-search">Limpar</button>
                </div>
            </div>
        </form>

        <div id="itens-list-container" class="overflow-auto" style="max-height: 300px;">
            <div id="itens-pagination-container" class="mt-3">
                </div>
        </div>
    <div class="modal-footer border-0 justify-content-end">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Fechar</button>
    </div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_solicitar_retirada.html

```

<div class="modal fade" id="modalSolicitarRetirada" tabindex="-1" aria-labelledby="modalSolicitarRetiradaLabel">
    <div class="modal-dialog modal-lg modal-dialog-centered">
        <div class="modal-content rounded-4 shadow">
            <div class="modal-header border-0">
                <h5 class="modal-title" id="modalSolicitarRetiradaLabel">Solicitar Retirada de Itens</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
            </div>
            <div class="modal-body">
                <form id="form-solicitar-retirada">
                    <div class="row g-3">
                        <div class="col-md-6">
                            <label for="solicitar_setor_id" class="form-label">Setor*</label>
                            <select class="form-select" id="solicitar_setor_id" name="setor_id" required>
                                <option value="" disabled selected>Selecione um setor</option>
                            </select>
                        </div>
                        <div class="col-md-6">
                            <label for="solicitado_localmente_por" class="form-label">Solicitado Localmente por*</label>
                            <input type="text" class="form-control" id="solicitado_localmente_por" name="solicitado_localmente_por">
                            <small class="form-text text-muted">Preencha se a solicitação for feita por alguém</small>
                        </div>
                        <div class="col-12">
                            <label for="solicitar_justificativa" class="form-label">Justificativa da Retirada</label>
                            <textarea class="form-control" id="solicitar_justificativa" name="justificativa"></textarea>
                        </div>
                    </div>

                    <hr class="my-4">
                    <h5>Itens para Retirada</h5>
                </form>
            </div>
        </div>
    </div>
</div>

```

```

<div id="itens-para-retirada-container" class="row g-3 mb-3">
  <div class="col-12 text-muted" id="no-items-message">Nenhum item adicionado
</div>

<div class="row g-3 align-items-end">
  <div class="col-12" id="selected-item-display" style="display: none;">
    <p class="mb-1">Item Selecionado: <strong id="selected-item-name"></strong>
  </div>
  <div class="col-md-6">
    <label for="quantidade_add_item" class="form-label"> Quantidade</label>
    <input type="number" class="form-control" id="quantidade_add_item" min="1"
  </div>
  <div class="col-md-6 d-flex align-items-end">
    <button type="button" class="btn btn-info w-100 me-2" id="btn-abrir-selecao">Abrir Seleção
    <button type="button" class="btn btn-success w-100" id="btn-adicionar-item">Adicionar Item
  </div>
</div>

</div>
</form>
</div>
<div class="modal-footer border-0 justify-content-center">
  <button type="button" class="btn btn-secondary me-2" data-bs-dismiss="modal">Cancelar</button>
  <button type="button" class="btn btn-primary" id="btn-salvar-solicitacao-retirada">Salvar Solicitação
</div>
</div>
</div>
</div>

```

Arquivo: C:\Users\Victor\Desktop\projeto_almoxarifado\sist_almoxarifado_ets\frontend\templates\partials\modals\modal_ver_detalhes_retirada.html

```

<div class="modal fade" id="modalVerDetalhesRetirada" tabindex="-1" aria-labelledby="modalVerDetalhesRetiradaLabel">
  <div class="modal-dialog modal-lg modal-dialog-centered">
    <div class="modal-content rounded-4 shadow">
      <div class="modal-header border-0">
        <h5 class="modal-title" id="modalVerDetalhesRetiradaLabel">Detalhes da Retirada</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Fechar"></button>
      </div>
      <div class="modal-body">
        <form class="row g-3">
          <div class="col-md-6">
            <label for="detalheRetiradaId" class="form-label">ID da Retirada</label>
            <input type="text" class="form-control" id="detalheRetiradaId" readonly>
          </div>
          <div class="col-md-6">
            <label for="detalheStatus" class="form-label">Status</label>
            <input type="text" class="form-control" id="detalheStatus" readonly>
          </div>
          <div class="col-md-6">
            <label for="detalheSetor" class="form-label">Setor Solicitante</label>
            <input type="text" class="form-control" id="detalheSetor" readonly>
          </div>
          <div class="col-md-6">
            <label for="detalheUsuario" class="form-label">Usuário Solicitante</label>
            <input type="text" class="form-control" id="detalheUsuario" readonly>
          </div>
          <div class="col-12">
            <label for="detalheJustificativa" class="form-label">Justificativa da Solicitação</label>

```

```

        <textarea class="form-control" id="detalheJustificativa" rows="2" readonly></textarea>
    </div>
    <div class="col-md-6">
        <label for="detalheSolicitadoPor" class="form-label">Solicitado Localmente Por</label>
        <input type="text" class="form-control" id="detalheSolicitadoPor" readonly>
    </div>
    <div class="col-md-6">
        <label for="detalheAutorizadoPor" class="form-label">Autorizado/Negado Por</label>
        <input type="text" class="form-control" id="detalheAutorizadoPor" readonly>
    </div>
    <div class="col-12">
        <label for="detalheData" class="form-label">Data da Solicitação</label>
        <input type="text" class="form-control" id="detalheData" readonly>
    </div>
    <div class="col-12">
        <label for="detalheStatusDesc" class="form-label">Detalhe do Status</label>
        <input type="text" class="form-control" id="detalheStatusDesc" readonly>
    </div>
    <div class="col-12">
        <label class="form-label">
            Itens Solicitados
            <small class="text-muted">(clique para ver detalhes)</small>
            <i class="bi bi-info-circle" data-bs-toggle="tooltip"
                title="Cada item é clicável para abrir detalhes"></i>
        </label>
        <div id="detalheItens" class="list-group"></div>
    </div>
</form>
</div>
<div class="modal-footer border-0 justify-content-center">
    <button class="btn btn-secondary" data-bs-dismiss="modal">Fechar</button>
</div>
</div>
</div>
</div>

```