Matheus Morais de Oliveira

Software de Gestão de Documentos

Passo Fundo - RS

10/2024

Resumo

Este trabalho apresenta o desenvolvimento de um Sistema de Gestão de Documentos integrado ao Active Directory, visando melhorar a eficiência e a segurança na gestão documental. A problemática abordada concentra-se nos desafios enfrentados pela falta de um sistema adequado, como controle insuficiente de acesso, ausência de versionamento e um fluxo de aprovação ineficaz, resultando em processos manuais suscetíveis a erros e vulnerabilidades de segurança. O objetivo geral é desenvolver um sistema que permita a autenticação de usuários via Active Directory, o cadastro e armazenamento seguro de documentos, a implementação de um fluxo de aprovação com múltiplos responsáveis e o registro de logs de acesso e ações realizadas. A metodologia aplicada baseou-se na utilização do framework Django em Python para o backend, integração com o Active Directory por meio da biblioteca ldap3 e banco de dados MySQL para armazenamento persistente. Adotou-se a metodologia ágil Scrum para o gerenciamento do projeto, permitindo entregas incrementais e feedback contínuo. Como resultado, obteve-se um sistema funcional que proporciona autenticação segura, controle de acesso baseado em permissões, um fluxo de aprovação robusto e registro completo de atividades, atendendo às necessidades identificadas. A implementação deste sistema contribui para maior segurança, eficiência operacional e facilidade de gerenciamento na organização.

Palavras-Chave: gestão de documentos; Active Directory; Django; autenticação LDAP.

1. INTRODUÇÃO

A gestão eficiente de documentos é um aspecto crítico para o sucesso operacional de organizações modernas. Com o aumento exponencial de informações e a necessidade de acesso rápido e seguro a documentos, torna-se indispensável implementar sistemas que facilitem o armazenamento, a recuperação e o controle de versões destes arquivos. No contexto da Engenharia de Software, a integração de sistemas de gestão de documentos com infraestruturas existentes, como o Active Directory (AD), apresenta-se como uma solução promissora para

centralizar a autenticação e autorizações, garantindo segurança e conformidade com as políticas corporativas.

Atualmente, muitas organizações enfrentam desafios significativos na gestão de seus documentos devido a processos manuais e fragmentados. A ausência de um sistema integrado resulta em dificuldades para controlar o acesso aos documentos, falta de versionamento adequado, riscos de acesso não autorizado e ineficiências no fluxo de aprovação. Esses problemas não apenas afetam a produtividade, mas também expõem a organização a riscos de segurança e não conformidade com regulamentos.

O **problema** central que este trabalho busca solucionar é a ineficiência e insegurança na gestão de documentos dentro da organização, decorrentes da ausência de um sistema integrado que permita o controle adequado de acesso, versionamento e aprovação de documentos, alinhado às práticas de segurança e autenticação já estabelecidas pela empresa, como o Active Directory.

O **objetivo geral** deste projeto é desenvolver um Sistema de Gestão de Documentos integrado ao Active Directory, visando aprimorar a eficiência operacional e a segurança na gestão documental da organização. Este sistema permitirá a centralização do controle de acesso, aproveitando a infraestrutura existente do AD para autenticação e autorização de usuários, além de proporcionar funcionalidades essenciais para o gerenciamento de documentos.

Para alcançar o objetivo geral, foram definidos os seguintes **objetivos específicos**:

- **Implementar autenticação integrada com o Active Directory**: Permitir que os usuários utilizem suas credenciais corporativas para acessar o sistema, garantindo segurança e facilitando o gerenciamento de permissões.
- **Desenvolver funcionalidades para cadastro e armazenamento seguro de documentos**: Criar interfaces e processos que possibilitem o upload, armazenamento e organização de documentos de forma estruturada.
- **Criar um fluxo de aprovação de documentos com múltiplos responsáveis**: Implementar um sistema de workflow que permita a revisão e aprovação de documentos por diferentes níveis hierárquicos ou setores responsáveis.

- **Implementar mecanismos de busca e filtragem avançada de documentos**: Facilitar a localização de documentos por meio de filtros por nome, categoria, data, entre outros critérios relevantes.
- **Registrar e monitorar logs de acesso e ações realizadas no sistema**: Garantir a rastreabilidade das ações dos usuários, aumentando a segurança e possibilitando auditorias internas.

A **justificativa** para o desenvolvimento deste software baseia-se na necessidade premente de aprimorar os processos internos relacionados à gestão de documentos. A integração com o Active Directory é fundamental, pois aproveita a infraestrutura de autenticação já estabelecida, reduzindo redundâncias e melhorando o controle de acesso. Além disso, a implementação de um sistema dedicado contribui para a padronização de processos, aumenta a eficiência operacional, reduz erros manuais e fortalece a segurança da informação. Em um ambiente corporativo onde a informação é um ativo valioso, garantir que os documentos estejam acessíveis de forma segura e organizada é essencial para a tomada de decisões e para a manutenção da competitividade da organização.

2. ESPECIFICAÇÕES INICIAIS DO SOFTWARE

2.1 Escopo do Produto

O sistema visa automatizar e garantir a segurança no gerenciamento de documentos corporativos. Os principais recursos incluem:

Autenticação Integrada: Acesso via Active Directory, centralizando e fortalecendo a autenticação.

Cadastro e Armazenamento: Cadastro de documentos, controle de revisões e armazenamento seguro.

Aprovação de Documentos: Fluxo de aprovação para revisão e validação por múltiplos responsáveis.

Visualização e Busca: Exibição de documentos aprovados com filtros de busca.

Registro de Ações: Logs detalhados de acessos e modificações, para auditoria e segurança.

Gestão de Usuários e Grupos: Importação do AD, criação de grupos e controle de permissões.

2.2 Funcionalidade do Produto

Login e Controle de Acesso: Autenticação por Active Directory e permissões específicas para usuários e grupos.

Cadastro e Versionamento de Documentos: Interface intuitiva para upload, versionamento e controle de documentos.

Fluxo de Aprovação: Definição de aprovadores com aprovação/rejeição documentada e histórico.

Busca e Visualização: Busca avançada e visualização dos documentos diretamente na plataforma.

Notificações Internas: Sistema de notificações informando aprovações e rejeições.

2.3 Ambiente Operacional e Tecnologias

O sistema é desenvolvido em Python e Django, com MySQL como banco de dados, e integra-se ao Active Directory via ldap3. Outras tecnologias incluem HTML5, CSS3 e JavaScript para o frontend, além de Aspose.Words para manipulação de documentos e geração de PDFs.

Requirements.txt

asgiref==3.8.1, aspose-words==24.10.0, Django==5.1.2, ldap3==2.9.1, mysqlclient==2.2.4, pyasn1==0.6.1, sqlparse==0.5.1, tzdata==2024.2.

Ambiente Operacional:

Compatível com servidores Windows e Linux, acessível via navegadores modernos.

3. METODOLOGIA

O desenvolvimento do sistema de gestão de documentos seguiu a metodologia ágil Scrum, ainda que sem a publicação formal de sprints, proporcionando um processo flexível e adaptável. Esta abordagem foi escolhida para permitir um desenvolvimento incremental e iterativo, com foco em entregas contínuas e ajustes rápidos às necessidades do projeto.

- **Planejamento e Organização**: O projeto foi organizado em ciclos curtos e incrementais, com reuniões para revisão de progresso e ajustes nos requisitos.
- **Ferramentas Utilizadas**: Utilizou-se Git para controle de versão, GitHub para repositório de código, e Kanban para organizar tarefas. O ambiente virtual em Python e testes automatizados foram implementados para assegurar qualidade e consistência no desenvolvimento.

Processo de Desenvolvimento: As etapas incluíram levantamento de requisitos, planejamento, design do sistema com diagramas UML, implementação iterativa, testes unitários e integrados, e uma implantação piloto para feedback.

A escolha pelo Scrum, mesmo sem sprints formais, foi motivada pela necessidade de flexibilidade, permitindo revisões contínuas e entregas de alta qualidade. As tecnologias principais incluem Django, MySQL, ldap3 para autenticação via Active Directory, e Aspose.Words para manipulação de documentos.

4. DESENVOLVIMENTO

4.1 Diagrama UML

Diagrama de Classes

Abaixo está o diagrama de classes, representando as principais entidades e relacionamentos no sistema:

- **Usuario**: representa os usuários que interagem com o sistema, podendo ser autenticados pelo Active Directory.
- **Grupo**: define grupos de usuários, possibilitando o gerenciamento de permissões coletivas.
- **Documento**: representa os documentos gerenciados no sistema, com atributos como nome, revisão e categoria.
- **Categoria**: classifica os documentos, permitindo uma organização estruturada.

- **Notificacao**: registra notificações para os usuários, como solicitações de aprovação.
- **Acesso**: armazena o histórico de acessos aos documentos, registrando usuário e data de visualização.

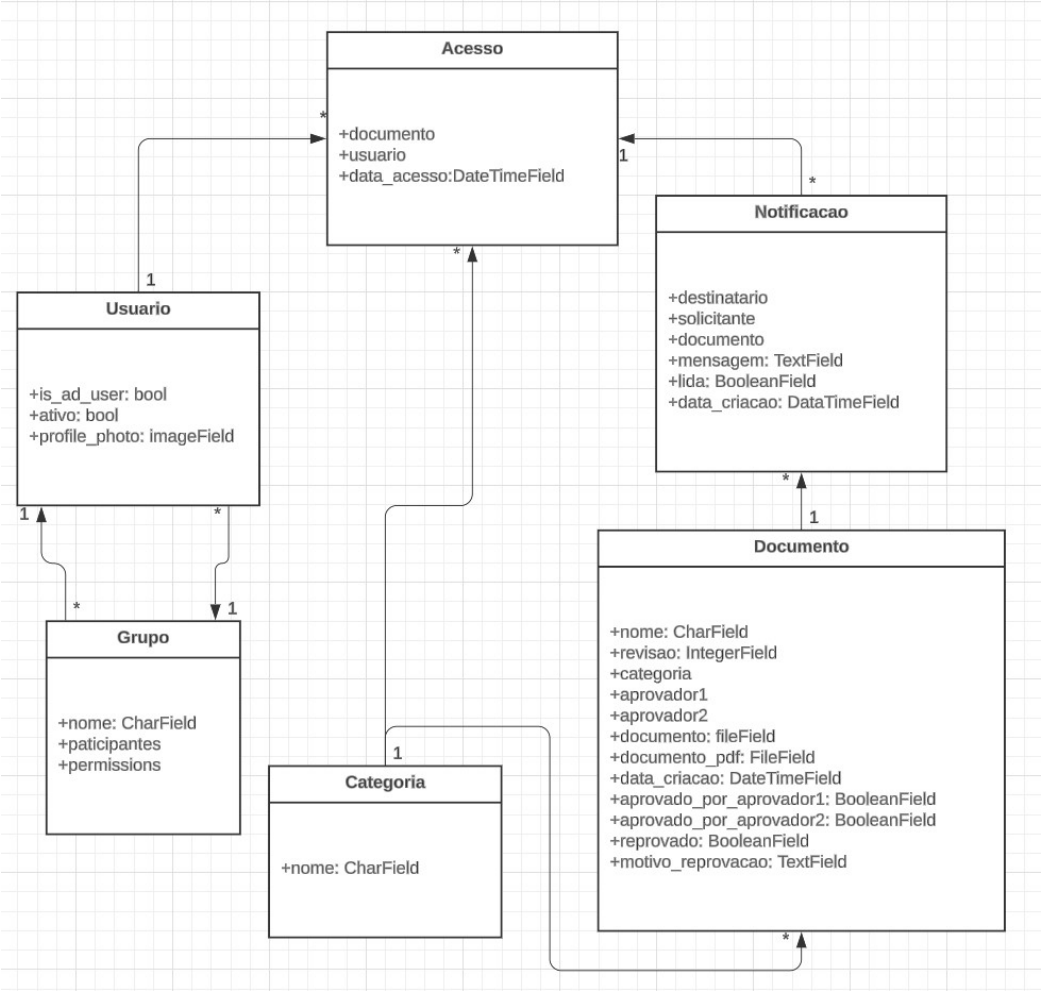

Figura 1- Diagrama de classes

Diagrama de Casos de Uso

O diagrama de casos de uso detalha as principais interações entre os atores e o sistema. Os atores incluem:

- **Usuário**: Pode realizar login, visualizar documentos, criar documentos, aprovar ou reprovar documentos, visualizar notificações e marcar notificações como lidas.

- **Administrador**: Responsável pela gestão de usuários e grupos, controle de permissões, e monitoramento de logs de acesso.

Casos de uso principais:

- **Login**: Autenticação com o Active Directory.
- **Criar Documentos**: Permite que usuários autorizados registrem novos documentos.
- **Aprovação de Documentos**: Fluxo de aprovação/reprovação realizado pelos usuários aprovadores.
- **Gestão de Usuários e Grupos**: Cadastro e administração de usuários e permissões.
- **Notificações:** Gerenciadas pelo sistema quando o usuário realiza alguma ação que dispara o sinal.

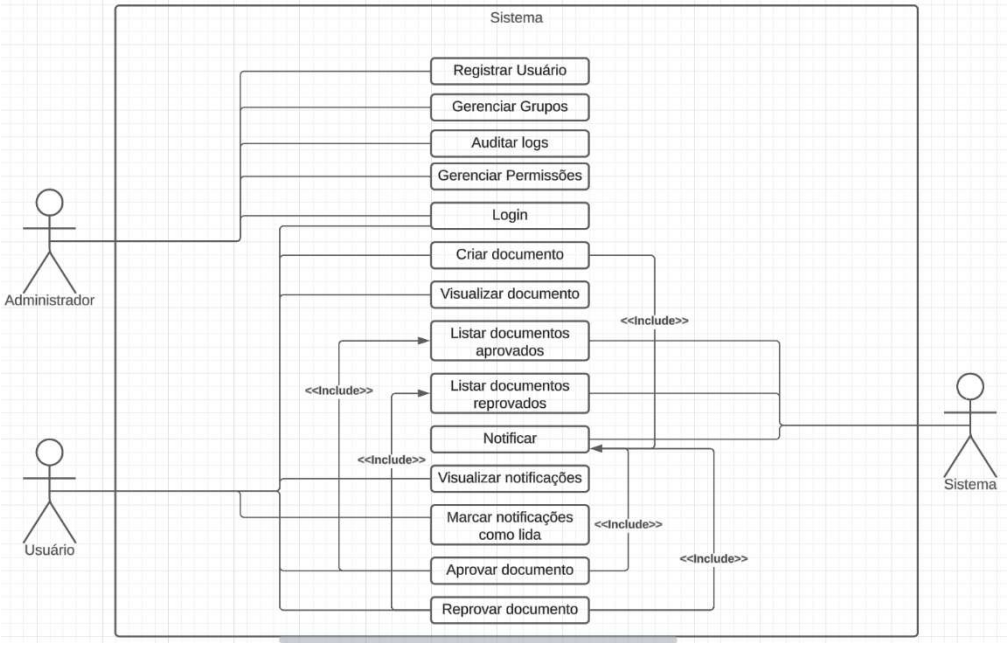

Figura 2 - Diagrama de casos de uso

4.3 Arquitetura do Software e Tecnologias Utilizadas

A arquitetura do sistema segue o padrão **MVC (Model-View-Controller)**, utilizando o framework Django para organizar a aplicação de forma modular.

- **Modelos**: Representam a estrutura dos dados e realizam a comunicação com o banco de dados.

- **Views**: Gerenciam a lógica do sistema e processam as requisições.
- **Templates**: Arquivos HTML que definem a interface do usuário.

Tecnologias Utilizadas:

- **Backend**: Python, Django e ldap3 para integração com Active Directory.
- **Banco de Dados**: MySQL, para armazenamento de dados estruturado.
- **Frontend**: HTML, CSS e JavaScript para construção das interfaces.
- **Controle de Versionamento**: Git e GitHub para versionamento e armazenamento de código.

Requisitos de Implantação:

- **Servidor**: Linux (Ubuntu ou CentOS) ou Windows Server.
- **Dependências**: Python 3.12+, Django, ldap3, Aspose.Words.
- **Navegadores**: Compatível com Google Chrome, Mozilla Firefox e Microsoft Edge.

4.4 Telas e Códigos Relevantes

As principais interfaces desenvolvidas incluem:

- **Tela de Login**: Autenticação via Active Directory.
- **Cadastro de Documentos**: Formulário para adicionar novos documentos, permitindo upload e categorização.
- **Aprovação de Documentos**: Tela para revisores aprovar/reprovar documentos.
- **Listagem de Documentos**: Exibe documentos aprovados, com filtros de busca.
- **Notificações**: Exibe notificações pendentes para os usuários.

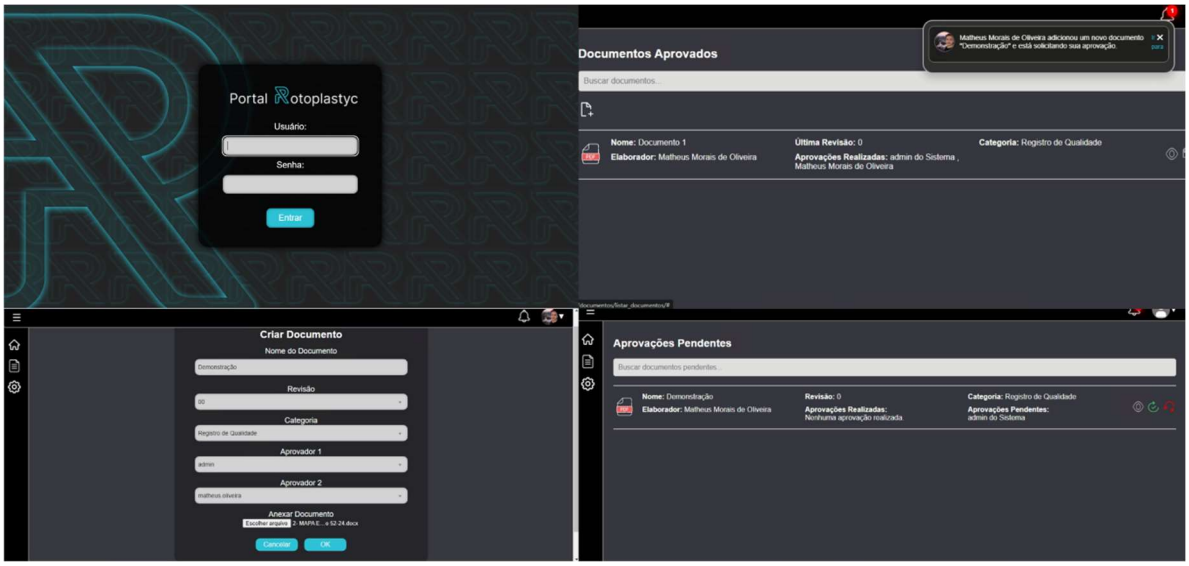

Figura 3 - Tela de login, Lista de documentos, Tela para criar novo documento, tela de aprovações do documento

Códigos Relevantes:

Exemplo de autenticação via Active Directory:

```python
def autenticar_usuario_ad(username, password):
    ldap_server = "ldap://tcc1.net"
    admin_user = "CN=Administrator,CN=Users,DC=tcc1,DC=net"
    admin_password = "Admin@ti"

    try:
        logger.info(f"Tentando autenticar o usuário {username} no LDAP.")
        server = Server(ldap_server, get_info=ALL)

        # Conexão com credenciais de administrador para buscar o DN do usuário
        admin_conn = Connection(server, user=admin_user, password=admin_password, au
        search_base = "CN=Users,DC=tcc1,DC=net"
        search_filter = f"(sAMAccountName={username})"
        admin_conn.search(search_base, search_filter, attributes=['distinguishedName

        if not admin_conn.entries:
            logger.warning(f"Usuário {username} não encontrado no LDAP.")
            admin_conn.unbind()
            return False

        user_dn = admin_conn.entries[0].distinguishedName.value
        admin_conn.unbind()

        # Tentar autenticar com as credenciais do usuário
        user_conn = Connection(server, user=user_dn, password=password, auto_bind=Tr
        logger.info(f"Usuário {username} autenticado com sucesso no LDAP.")
        user_conn.unbind()
        return True

    except Exception as e:
        logger.error(f"Erro ao autenticar o usuário {username} no LDAP: {str(e)}")
        return False
```

Figura 4 - Trecho do código de autenticação de usuários no AD

Código para criação de um novo documento e atribuição de aprovador:

```python
@login_required
@permission_required('documentos.add_documento', raise_exception=True)
def criar_documento(request):
    logger.debug(f"[criar_documento] Iniciando processo de criação de documento para o usuário: {request.user.username}")

    if request.method == 'POST':
        logger.debug("[criar_documento] Recebendo dados do formulário POST")
        form = DocumentoForm(request.POST, request.FILES)

        if form.is_valid():
            logger.debug("[criar_documento] Formulário válido. Tentando salvar o documento.")
            try:
                with transaction.atomic():
                    documento = form.save(commit=False)  # Não salvar imediatamente
                    documento.elaborador = request.user  # Atribuir o elaborador manualmente
                    documento.save()  # Agora salva o documento com o elaborador atribuído
                    logger.info(f"[criar_documento] Documento criado com sucesso: {documento.nome} - Revisão {documento.revisao}")
                    messages.success(request, 'Documento criado com sucesso!')
                    return redirect('documentos:listar_documentos_aprovados')
            except Exception as e:
                logger.error(f"[criar_documento] Erro ao salvar documento: {e}", exc_info=True)
                messages.error(request, f'Ocorreu um erro ao criar o documento: {e}')
        else:
            logger.warning(f"[criar_documento] Formulário inválido. Erros: {form.errors}")
    else:
        logger.debug("[criar_documento] Requisição GET recebida. Renderizando formulário vazio.")
        form = DocumentoForm()

    return render(request, 'documentos/criar_documento.html', {'form': form})
```

Figura 5 - Trecho do código para criação de um novo documento

4.5 Repositório do Código-Fonte

O código completo do sistema de gestão de documentos está disponível em um repositório no GitHub, facilitando o acesso e a colaboração no desenvolvimento do projeto.

Link do Repositório: https://github.com/matheusmorais31/Portal_Rotoplastyc

5. CONSIDERAÇÕES FINAIS

O desenvolvimento do Sistema de Gestão de Documentos integrado ao Active Directory alcançou os objetivos propostos, proporcionando uma solução eficiente e segura para a administração de documentos na organização. A integração com o Active Directory permitiu autenticação centralizada, garantindo que apenas usuários autorizados acessem o sistema, o que reforça a segurança e facilita a gestão de permissões.

As funcionalidades implementadas, como cadastro, aprovação, visualização e busca de documentos, além do registro detalhado de acessos, contribuíram para a melhoria dos processos internos, reduzindo erros e aumentando a eficiência operacional. A utilização de tecnologias como Python, Django, MySQL e integração com o Active Directory demonstrou-se adequada para atender às necessidades do projeto, oferecendo uma base sólida e escalável para futuras expansões.

Para trabalhos futuros, sugere-se a implementação de recursos adicionais, como controle de editáveis, analise intermediaria, notificações por e-mail entre outras melhorais para facilitar o fluxo e manter tudo centralizado dentro da aplicação.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, Cristiano. Como utilizar o módulo PythonLDAP. Disponível em: https://wiki.python.org.br/PythonLdap. Acesso em: 26 set. 2008.

EQUIPE TOTVS. JWT token: o que é, estrutura e as vantagens de usar. Disponível em: https://www.totvs.com/blog/gestao-para-assinatura-de-documentos/jwt-token. Acesso em: 11 abr. 2024.

FILHO, Luiz Cezar Medeiros. O que é e como funciona a Autenticação Integrada?. Disponível em: https://www.neomind.com.br/blog/o-que-e-e-como-funciona-a-autenticacao-integrada/. Acesso em: 7 maio 2019.

MICROSOFT. Como configurar e usar a Integração do Active Directory para atribuição de agentes. Disponível em: https://learn.microsoft.com/pt-br/system-center/scom/manage-ad-integration-agent-assignment?view=sc-om-2022. Acesso em: 18 abr. 2024.

ORACLE. Documentação MySQL. Disponível em: https://docs.oracle.com/en-us/iaas/mysql-database/doc/getting-started.html. Acesso em: 24 jul. 2023.