Documentação: Sistema de Gerência de Dispositivos de TI

Gabriel Lima Scheffler

Analista de Dados

1. Introdução

O Sistema de Gerência de Dispositivos é uma plataforma centralizada, desenvolvida especificamente para atender às demandas críticas e ao ambiente dinâmico do setor de TI em um contexto hospitalar. Em um cenário onde a disponibilidade e a performance dos equipamentos tecnológicos estão diretamente ligadas à qualidade do atendimento e à segurança do paciente, ter um controle preciso dos ativos de TI não é apenas uma necessidade administrativa, mas uma exigência operacional.

O software funciona como um banco de informações rápidas e consolidadas, oferecendo uma visão 360 graus sobre todos os dispositivos do hospital. Além do inventário, a plataforma integra módulos essenciais para a gestão de treinamentos de equipe, planos de manutenção preventiva e corretiva, e um sistema robusto de chamados (tickets) para suporte técnico ágil e documentado.

O principal objetivo do sistema é transcender o simples gerenciamento, consolidando-se como uma central de *analytics* para o setor de TI. Através dele, é possível gerar relatórios precisos sobre a qualidade de serviço (QoS), medindo tempos de resposta, resolução de problemas e a eficiência da equipe. Um dos seus maiores diferenciais é a capacidade de analisar a taxa de depreciação dos equipamentos e, em fases futuras, aplicar algoritmos de predição e regressão. Essa abordagem proativa permitirá identificar a necessidade de futuras manutenções antes que as falhas ocorram, minimizando o tempo de inatividade de sistemas críticos e otimizando os investimentos em tecnologia.

É importante ressaltar que o projeto se encontra atualmente em fase beta. As funcionalidades essenciais estão operacionais, mas o sistema está em contínuo desenvolvimento, com aprimoramentos e a implementação de funcionalidades avançadas, como os algoritmos preditivos, ainda em andamento. O objetivo final é transformar a gestão de TI de um modelo reativo para um modelo preditivo e orientado por dados, garantindo a máxima continuidade operacional para o hospital.

2. Requisitos do Projeto

Esta seção detalha os requisitos funcionais e não funcionais que norteiam o desenvolvimento e a operação do Sistema de Gerência de Dispositivos.

2.1. Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades e comportamentos que o sistema deve executar.

- Módulo de Gestão de Ativos (Inventário Inteligente)
    - Cadastro Completo: Permitir o cadastro de dispositivos com informações detalhadas: tipo (computador, servidor, impressora e roteador), marca, modelo, número de série, data de aquisição, data de fim da garantia, e fornecedor.
    - Controle do Ciclo de Vida: Gerenciar e rastrear o status de cada ativo (Em estoque, em uso, em manutenção, Descartado).
    - Associação de Ativos: Associar dispositivos a usuários, setores e locais físicos (ex: Sala de Cirurgia 1, Recepção), permitindo uma localização rápida.
    - Histórico de Ativos: Manter um log completo de todas as alterações, manutenções, chamados e usuários associados a um dispositivo ao longo de sua vida útil.
    - Gestão de Licenças de Software: Cadastrar e associar licenças de software aos dispositivos, controlando o uso e as datas de expiração.
- Módulo de Gestão de Usuários e Setores
    - Cadastro de Colaboradores: Manter um registro centralizado de todos os funcionários, incluindo cargo, setor e informações de contato.
    - Estrutura Organizacional: Permitir a criação e gestão da estrutura de setores/departamentos do hospital.
- Módulo de Suporte e Chamados (Ticketing Avançado)
    - Abertura Simplificada de Chamados: Permitir que usuários abram chamados através de um portal simples.
    - Priorização e Categorização: Classificar chamados por categoria (ex: Hardware, Software, Rede) e nível de prioridade (ex: Crítico, Alto, Médio, Baixo).

    - Gestão de SLA (Acordo de Nível de Serviço): Definir e monitorar tempos de resposta e solução para cada prioridade, com alertas visuais para chamados próximos do vencimento.
    - Fluxo de Trabalho de Atendimento: Atribuir chamados a técnicos ou equipes específicas e registrar todas as interações e soluções aplicadas.
    - Base de Conhecimento: Permitir que técnicos criem artigos a partir de chamados resolvidos, facilitando a solução de problemas recorrentes.
- Módulo de Manutenção
    - Plano de Manutenção Preventiva: Agendar e gerenciar manutenções recorrentes para equipamentos críticos, gerando ordens de serviço automaticamente.
    - Registro de Manutenção Corretiva: Documentar todas as manutenções não planejadas, associando custos de peças e serviços ao ativo correspondente.
- Módulo de Analytics e Predição
    - Dashboard Analítico: Apresentar um painel visual com KPIs (Indicadores Chave de Desempenho) em tempo real: total de chamados abertos/fechados, tempo médio de atendimento (TMA), taxa de resolução no primeiro contato, e status geral dos ativos.
    - Cálculo de Depreciação: Calcular automaticamente a depreciação dos ativos com base na data de compra e vida útil estimada.
    - Geração de Relatórios Gerenciais: Exportar relatórios customizáveis em PDF sobre inventário, conformidade de licenças, custos de manutenção e performance da equipe de TI.
    - Análise Preditiva (Requisito Futuro): Desenvolver a capacidade de analisar o histórico de chamados e a idade dos equipamentos para gerar um "score de risco" de falha, sinalizando dispositivos que necessitam de atenção proativa.
- Módulo de Administração e Integração
    - Autenticação e Permissões (RBAC): Implementar um controle de acesso baseado em papéis (Role-Based Access Control), definindo perfis como Administrador, Técnico de TI e Usuário Final, cada um com permissões específicas.

    - Alertas e Notificações: Enviar notificações automáticas por e-mail ou no sistema sobre eventos importantes (novo chamado atribuído, SLA próximo de vencer, garantia expirando).
    - API de Integração: Oferecer endpoints RESTful seguros e documentados para a comunicação com o agente de monitoramento e futuras integrações com outros sistemas hospitalares.

2.2. Requisitos Não Funcionais

Os requisitos não funcionais definem os critérios de qualidade e operação do sistema.

- Desempenho
    - Tempo de Resposta: As páginas e consultas críticas do sistema devem carregar em menos de 3 segundos em condições normais de uso.
    - Responsividade: A interface web deve ser totalmente funcional e adaptável a diferentes tamanhos de tela (desktops, tablets).
- Segurança
    - Criptografia de Dados: Todo o tráfego entre o cliente e o servidor deve ser criptografado via HTTPS (SSL/TLS). Dados sensíveis no banco de dados devem ser armazenados de forma segura.
    - Logs de Auditoria: O sistema deve registrar todas as ações críticas (login, alterações de dados, exclusões) com informações de usuário, data e hora, para fins de auditoria e segurança.
    - Proteção contra Vulnerabilidades: O sistema deve ser desenvolvido seguindo as melhores práticas de segurança para prevenir vulnerabilidades comuns (ex: SQL Injection, XSS).
- Confiabilidade e Disponibilidade
    - Alta Disponibilidade: O sistema deve ser projetado para ter um tempo de atividade (uptime) de no mínimo 99.5%, dado sua importância no ambiente hospitalar.
    - Backup e Recuperação: Deve existir uma política de backups automáticos e regulares do banco de dados, com um plano de recuperação de desastres testado.
- Escalabilidade
    - Escalabilidade Vertical e Horizontal: A arquitetura deve permitir o aumento de recursos do servidor (vertical) e, se necessário, a adição de

mais servidores (horizontal) para suportar o crescimento do número de ativos e usuários.

- Manutenibilidade e Usabilidade
    - Código Limpo e Documentado: O código-fonte deve ser modular, bem comentado e seguir as convenções do Django para facilitar a manutenção e a integração de novos desenvolvedores.
    - Interface Intuitiva: A interface do usuário deve ser limpa, consistente e de fácil aprendizado, reduzindo a necessidade de treinamento extensivo para os usuários finais.

3. Arquitetura do Software

A arquitetura do Sistema de Gerência de Dispositivos foi projetada para ser robusta, escalável e de fácil manutenção, baseando-se em princípios de separação de responsabilidades. Ela adota um modelo Cliente-Servidor distribuído e é pensada para uma implantação moderna via contêineres.

Os componentes principais são:

3.1. Aplicação Web (Backend - Servidor Django)

O núcleo do sistema é uma aplicação monolítica, porém modular, construída com o framework Django. Ele é responsável por toda a lógica de negócios, processamento de dados e interação com o usuário. Sua estrutura interna segue o padrão Model-View-Template (MVT):

- Model (ORM do Django): Esta camada é a representação dos dados. Utilizando o Object-Relational Mapper (ORM) do Django, as tabelas do banco de dados PostgreSQL são mapeadas para classes Python. Isso abstrai a complexidade do SQL, aumenta a segurança (prevenindo SQL injection) e agiliza o desenvolvimento.
- View (Lógica de Negócios): As views processam as requisições HTTP recebidas. Elas orquestram a interação entre os Models (para buscar ou salvar dados) e os Templates (para renderizar a resposta). Aqui reside a lógica principal do sistema: validações, cálculos para relatórios, controle de permissões e a exposição dos endpoints da API RESTful (provavelmente utilizando Django REST Framework) para o agente de monitoramento.
- Template (Camada de Apresentação): Composto por arquivos HTML, CSS e JavaScript, esta camada é responsável por gerar a interface que o usuário final vê e interage no navegador.

Para produção, o servidor Django opera através de uma interface WSGI (Web Server Gateway Interface), com um servidor de aplicação como o Gunicorn, que gerencia os processos e requisições de forma eficiente.

3.2. Banco de Dados (Persistência - PostgreSQL)

A escolha do PostgreSQL como sistema de gerenciamento de banco de dados (SGBD) é estratégica, devido aos seguintes fatores:

- Robustez e Conformidade ACID: Garante a integridade transacional dos dados, o que é crítico para operações como registrar um chamado e vinculá-lo a um ativo simultaneamente.

- Escalabilidade: Suporta grandes volumes de dados e um alto número de conexões concorrentes, permitindo que o sistema cresça junto com o inventário do hospital.
- Recursos Avançados: Oferece suporte nativo a tipos de dados complexos (como JSONB, ideal para armazenar dados de hardware não estruturados enviados pelo agente), consultas geoespaciais e busca de texto completo, que podem ser aproveitados em futuras funcionalidades.

3.3. Mecanismo de Monitoramento (Cliente - Agente Desktop)

O agente é um software leve, desenvolvido em CustomTkinter, projetado para ser executado em segundo plano nas estações de trabalho. Sua arquitetura interna consiste em:

- Módulo Coletor: Utiliza bibliotecas de sistema (como psutil em Python) para extrair informações de hardware em tempo real: uso de CPU, memória RAM total e em uso, espaço em disco, atividade de rede, etc.
- Módulo de Comunicação (Cliente API): Formata os dados coletados (geralmente em formato JSON) e os envia periodicamente para a API RESTful do servidor Django através de requisições HTTP POST seguras. A comunicação é autenticada, geralmente por meio de um token de API, para garantir que apenas agentes autorizados possam enviar dados.
- Agendador (Scheduler): Um componente interno que controla a frequência da coleta e envio de dados (ex: a cada 5 minutos), para não sobrecarregar nem o cliente, nem o servidor.

3.4. Conteinerização e Implantação (Docker)

Para garantir consistência entre os ambientes de desenvolvimento, teste e produção, e para simplificar a implantação, a arquitetura prevê a conteinerização de seus componentes usando Docker.

- Imagens Docker: São criadas imagens separadas para a aplicação Django e para o banco de dados PostgreSQL. Isso isola as dependências e configurações de cada serviço.
- Docker Compose: Em ambiente de desenvolvimento e em implantações mais simples, o docker-compose.yml é utilizado para orquestrar os contêineres. Ele define os serviços (web, db), suas configurações, volumes de dados e a rede interna pela qual eles se comunicam.
- Escalabilidade em Produção: Em um ambiente hospitalar crítico que exige alta disponibilidade, essa arquitetura conteinerizada permite uma fácil migração para orquestradores mais robustos como o Kubernetes (K8s), que pode

gerenciar a escalabilidade, o balanceamento de carga e a recuperação de falhas dos contêineres de forma automática.

4. Interação Humano-Computador (IHC)

A concepção da Interação Humano-Computador (IHC) do sistema foi guiada por um princípio central: eficiência operacional para a equipe de TI. A interface se adapta ao perfil do usuário, oferecendo a complexidade necessária para os técnicos e a clareza essencial para os usuários finais.

A interação se divide em duas experiências distintas:

4.1. Interface Web (Plataforma Central de Gestão)

Esta é a principal ferramenta de trabalho para a equipe de TI (gestores e técnicos). O design foca em apresentar uma alta densidade de informação de forma organizada e acessível, otimizando os fluxos de trabalho diários.

- Navegação e Estrutura da Informação:
    - Layout Consistente: O sistema utiliza uma barra de navegação lateral persistente que agrupa as principais funcionalidades (Dashboard, Ativos, Chamados, Usuários, Relatórios, etc.), permitindo acesso rápido a qualquer módulo a partir de qualquer página.
    - Hierarquia Visual Clara: Títulos, subtítulos e o uso de "breadcrumbs" (caminho de navegação) ajudam o usuário a se localizar e entender onde está dentro da estrutura do sistema.
    - Busca e Filtragem: Para lidar com grandes volumes de dados, são implementadas barras de pesquisa globais e filtros contextuais em todas as tabelas, permitindo que o técnico encontre um dispositivo ou chamado específico em segundos.
- Dashboard de Analytics (O Centro de Comando):
    - Visão Imediata: A primeira tela após o login é o dashboard, projetado para fornecer uma visão panorâmica e imediata do estado da infraestrutura de TI.
    - Componentes Interativos: A interação é feita através de:
        - Widgets: Pequenos blocos de informação (cards) que destacam KPIs importantes (ex: "Chamados Críticos Abertos", "Ativos Próximos do Fim da Garantia").
        - Gráficos Dinâmicos: Gráficos de barra, pizza e linha (gerados com Chart.js) que respondem à interação do usuário, exibindo informações detalhadas ao passar o mouse (tooltips).

        - Funcionalidade de *Drill-Down*: O usuário pode clicar em um segmento de um gráfico (ex: a fatia de "Notebooks" em um gráfico de tipos de ativos) para ser levado diretamente à lista filtrada desses itens.

- Fluxos de Trabalho de Gerenciamento (CRUD Otimizado):
    - Formulários Inteligentes: Os formulários de cadastro e edição são projetados para serem claros, com campos agrupados logicamente. A validação de dados ocorre em tempo real para prevenir erros de entrada antes do envio.
    - Tabelas de Dados Ricas: As listas de ativos, usuários e chamados são apresentadas em tabelas paginadas e ordenáveis, permitindo que o usuário organize a visualização conforme sua necessidade.
    - Ações Contextuais: Botões de ação (Adicionar Novo, Editar, Visualizar Detalhes, Excluir) são posicionados de forma consistente e, em alguns casos, ficam disponíveis diretamente na linha da tabela, agilizando as operações.
- Portal de Abertura de Chamados:
    - Os técnicos da Ti podem realizar o cadastro de chamados de forma retroativa sendo seu foco para instituições de menor porte, onde não tem a agilidade de uma equipe muito grande para agir com uma central de chamados entre o usuário final e Técnico

4.2. Agente Desktop (Monitoramento Não Intrusivo)

A filosofia de design do agente é "menos é mais". Sua função é coletar dados sem interferir no trabalho do usuário, que pode estar em uma atividade crítica.

- Interface Gráfica Mínima:
    - Operação em Segundo Plano: O agente é projetado para rodar silenciosamente em segundo plano, sem janelas ou pop-ups que possam distrair o usuário.
    - Ícone na Bandeja do Sistema: A única presença visual constante é um ícone na bandeja do sistema (system tray). O estado do ícone (ex: verde para conectado, vermelho para erro) fornece um feedback visual imediato sobre o status do monitoramento.
- Interações Possíveis:
    - Menu de Contexto: Ao clicar com o botão direito no ícone, um menu simples oferece poucas opções:

        - Status: Abre uma pequena janela com informações básicas ("Conectado ao servidor", "Última sincronização: 17:25").
        - Forçar Sincronização: Uma ação para enviar os dados de hardware imediatamente, útil para diagnóstico.
        - Abrir um Chamado: Um atalho que abre o portal web de chamados no navegador padrão.
    - Feedback Passivo: A principal "interação" do usuário com o agente é a confiança de que ele está funcionando. A ausência de alertas ou erros é, por si só, a melhor interface. As notificações são reservadas apenas para falhas críticas na comunicação com o servidor.

5. Ferramentas Utilizadas

A seleção de tecnologias para este projeto foi pautada em critérios de robustez, escalabilidade, segurança e agilidade no desenvolvimento. A seguir, detalhamos o papel de cada componente do ecossistema tecnológico.

5.1. Framework Backend

- Django:
    - Descrição: É um framework Python de alto nível que segue a filosofia "batteries-included" (baterias inclusas).
    - Papel no Projeto: Django é a espinha dorsal de toda a aplicação web. Ele não apenas gerencia as rotas (URLs) e requisições, mas também fornece componentes cruciais que aceleram o desenvolvimento:
    - ORM (Object-Relational Mapper): Permite que os desenvolvedores interajam com o banco de dados usando código Python, eliminando a necessidade de escrever SQL manualmente, o que aumenta a produtividade e a segurança.
    - Sistema de Autenticação: Oferece um sistema completo e seguro para gerenciamento de usuários, senhas, grupos e permissões, que é a base do controle de acesso do sistema.
    - Painel de Administração: Gera uma interface administrativa pronta para uso, ideal para a gestão básica dos dados do sistema sem a necessidade de desenvolver telas específicas.
    - Justificativa da Escolha: Foi escolhido por sua maturidade, ecossistema robusto, foco em segurança (com proteções nativas contra ataques como CSRF e XSS) e pela sua capacidade de escalar para aplicações complexas.

5.2. Banco de Dados

- PostgreSQL:
    - Descrição: Um sistema de gerenciamento de banco de dados objeto-relacional de código aberto, renomado por sua confiabilidade e vasta gama de funcionalidades.
    - Papel no Projeto: É o repositório central de todos os dados da aplicação, desde o inventário de dispositivos e registros de usuários até o histórico de chamados e logs de manutenção.

- Justificativa da Escolha: PostgreSQL foi selecionado por sua forte aderência aos padrões SQL, sua robustez em ambientes transacionais (conformidade ACID) e sua excelente performance com consultas complexas, essenciais para a geração de relatórios e análises do dashboard.
- psycopg2-binary:
- Descrição: É o adaptador (driver) de banco de dados PostgreSQL para Python.
- Papel no Projeto: Atua como a ponte de comunicação indispensável entre o Django e o PostgreSQL. Ele traduz as chamadas do ORM do Django em comandos que o banco de dados PostgreSQL entende e vice-versa.

5.3. GUI Desktop (Agente de Monitoramento)

- CustomTkinter:
  - Descrição: Uma biblioteca Python baseada no Tkinter, mas com um conjunto de widgets modernos e customizáveis.
  - Papel no Projeto: É utilizada para construir a interface gráfica mínima do agente de monitoramento que roda nas máquinas dos usuários. Permite criar uma janela de status visualmente agradável e de baixo consumo de recursos.
  - Justificativa da Escolha: Ideal para este caso de uso por ser leve, puramente em Python (facilitando a distribuição) e por oferecer uma aparência mais moderna que o Tkinter tradicional, sem a complexidade de frameworks de GUI mais pesados.

5.4. Bibliotecas de Análise e Relatórios

- Pandas:
  - Descrição: Uma biblioteca fundamental para manipulação e análise de dados em Python.
  - Papel no Projeto: No backend, o pandas é a ferramenta de "trabalho pesado" para os dados. Antes de exibir informações no dashboard ou gerar um PDF, o pandas é utilizado para agregar, filtrar e calcular métricas complexas (ex: tempo médio de resolução de chamados, contagem de ativos por status), transformando dados brutos do banco em insights valiosos.
  - plotly:

- Descrição: Uma biblioteca de visualização de dados que permite criar gráficos interativos de alta qualidade.
- Papel no Projeto: Enquanto o Chart.js roda no cliente, o Plotly pode ser usado no servidor (backend) para gerar imagens de gráficos mais complexos ou customizados que podem ser embutidos nos relatórios em PDF, oferecendo visualizações ricas mesmo em um formato estático.
- xhtml2pdf:
- Descrição: Uma biblioteca que converte conteúdo HTML e CSS em documentos PDF.
- Papel no Projeto: É o motor por trás da funcionalidade "Exportar para PDF". O sistema primeiro renderiza um template HTML com os dados e gráficos do relatório; em seguida, o xhtml2pdf processa esse HTML para gerar um arquivo PDF formatado e profissional, pronto para ser baixado.

5.5. Tecnologias Frontend

- Bootstrap:
  - Descrição: É um dos frameworks de código aberto mais populares do mundo para o desenvolvimento de componentes de interface e frontend.
  - Papel no Projeto: É a fundação do design visual e da responsividade da aplicação web. Bootstrap fornece um poderoso sistema de *grid* que permite que a interface se adapte perfeitamente a diferentes tamanhos de tela (desktops, tablets e celulares). Além disso, oferece uma vasta biblioteca de componentes pré-estilizados (botões, formulários, tabelas, modais, alertas) que garantem uma aparência consistente e profissional em todo o sistema, acelerando drasticamente o tempo de desenvolvimento da interface do usuário.
  - Justificativa da Escolha: Foi escolhido por sua maturidade, vasta documentação, enorme comunidade e pela capacidade de criar rapidamente interfaces limpas e responsivas, permitindo que a equipe de desenvolvimento se concentre mais na lógica de negócios do backend.

6. Estrutura do Banco de Dados

A estrutura do banco de dados é definida através do ORM (Object-Relational Mapper) do Django no arquivo dispositivos/models.py. O modelo é relacional e projetado para centralizar todas as informações de ativos, operações e usuários. Uma característica notável é o uso de Meta.managed = False na maioria dos modelos, indicando que o Django não gerencia a criação ou alteração das tabelas, mas apenas mapeia as classes para um esquema de banco de dados preexistente.

A estrutura pode ser dividida nos seguintes grupos de entidades:

6.1. Entidades Principais (Core)

Este grupo representa os principais objetos de negócio do sistema.

- Gestão de Ativos:
    - Dispositivos: Atua como uma tabela centralizadora e genérica para todos os tipos de ativos. Ela utiliza relações OneToOneField para se conectar a tabelas mais específicas. Essa abordagem permite que um chamado ou um plano de manutenção seja associado a um "dispositivo" genérico, independentemente de ser um computador, servidor etc.
    - Computadores, Servidores, Roteadores, Impressoras: São tabelas especializadas que armazenam os atributos específicos de cada tipo de ativo. Por exemplo, Servidores possui campos como service_tag, enquanto impressoras armazena o modelo do toner. Cada uma delas está ligada a um único registro na tabela Dispositivos.
- Estrutura Organizacional:
    - Setores: Tabela simples que define os departamentos do hospital (ex: "UTI", "Recepção", "TI"), usada para agrupar usuários e ativos.
    - Usuarios: Armazena as informações dos colaboradores do hospital (nome, função, CPF). É importante notar que este modelo parece ser distinto do AuthUser do Django, servindo para identificar os responsáveis pelos ativos e solicitantes de chamados.
- Módulos Operacionais:
    - Chamados: Uma das tabelas mais importantes, registrando cada ticket de suporte. Ela se conecta a múltiplas entidades através de chaves estrangeiras (ForeignKey), ligando um chamado a um Dispositivo específico, a um Usuario solicitante e a um Setor. Também contém campos para status, datas e descrição do problema.

- PlanoManuPrevent: Tabela para agendar e controlar as manutenções preventivas. Cada registro está associado a um Dispositivo, permitindo rastrear o histórico de manutenção de cada ativo.
- Treinamentos: Registra os eventos de treinamento oferecidos, associando-os a um Setor.

6.2. Entidades de Suporte e Configuração

Este grupo inclui tabelas que apoiam as entidades principais ou armazenam dados de configuração.

- CategoriasAtend: Uma tabela de apoio para classificar os tipos de chamados (ex: "Problema de Hardware", "Instalação de Software").
- Hosts: Armazena uma lista de hosts de rede com seus respectivos IPs.
- LoginUsuarioPc: Tabela que associa credenciais de login de máquina a um Computador específico.
- EmailsAntigos, EmailsNovos, PastaPublica: Tabelas para armazenar informações legadas ou credenciais de acesso a recursos compartilhados.

6.3. Entidade de Monitoramento

- MonitoramentoHardware: Tabela projetada para receber os dados enviados pelo agente desktop. Ela armazena séries temporais de métricas de hardware (cpu_percent, memory_percent, disk_percent) para cada Computador, com um timestamp para cada leitura. Essa tabela é a base para análises de desempenho e futuras predições de falhas.

6.4. Modelos Internos do Django

O arquivo também lista modelos que são padrão do framework Django e essenciais para seu funcionamento, mas não diretamente relacionados à lógica de negócio do hospital.

- Grupo de Autenticação (AuthUser, AuthGroup, AuthPermission etc.): Gerenciam os usuários que podem fazer login no sistema, seus grupos e as permissões de acesso que possuem. AuthUser é quem efetivamente acessa a plataforma web.
- Grupo de Administração (DjangoAdminLog, DjangoContentType etc.): Tabelas internas que o Django usa para o painel de administração, sessões de usuário (DjangoSession) e para controlar o estado das migrações de banco de dados (DjangoMigrations).

7. Estrutura do Servidor Django

A organização do projeto segue as convenções do Django para modularidade e clareza:

- **config/**: Contém as configurações globais do projeto, como settings.py (configuração do banco de dados, apps instalados, chaves secretas) e urls.py (roteamento principal de URLs).
- **dispositivos/**: É o aplicativo principal do projeto e concentra toda a lógica de negócio.
    - models.py: Define a estrutura do banco de dados.
    - views.py: Contém a lógica para processar requisições, interagir com o banco e renderizar páginas.
    - forms.py: Define os formulários HTML para criação e edição de registros.
    - templates/: armazena os arquivos HTML que compõem a interface do usuário.
    - urls.py: Define as rotas de URL específicas deste aplicativo.
    - utils.py: Funções auxiliares reutilizáveis.
- **static/**: armazena arquivos estáticos como CSS, JavaScript e imagens.
- **manage.py**: Utilitário de linha de comando para interagir com o projeto (ex: runserver, migrate).

8. Módulos do Sistema e Suas Funções

As funcionalidades do sistema são organizadas em módulos lógicos, diretamente acessíveis através da barra de navegação principal. Cada módulo é projetado para atender a um conjunto específico de tarefas do fluxo de trabalho de um departamento de TI. A lógica de cada módulo é implementada principalmente nas views (dispositivos/views.py), que interagem com os modelos do banco de dados para manipular e apresentar os dados.

A seguir, uma descrição de cada módulo presente na interface:

Ótima adição! Isso torna o dashboard ainda mais poderoso. Vou reescrever a seção para refletir com precisão essas funcionalidades de monitoramento em tempo real e relatórios consolidados.

8.1. Módulo Dashboard

O Dashboard é a tela inicial e o centro de comando do sistema, projetado para fornecer uma consciência situacional completa e em tempo real sobre a saúde da infraestrutura de TI do hospital. Ele consolida dados críticos de múltiplos módulos em uma única interface visual e interativa.

- Funções Principais:
    - Monitoramento de Hardware em Tempo Real: Exibe gráficos dinâmicos que mostram o consumo de CPU, memória RAM e uso de disco (HDD/SSD) dos computadores monitorados. Isso permite à equipe de TI identificar proativamente máquinas com baixo desempenho ou problemas de recursos antes que afetem os usuários.
    - Relatório Consolidado de Manutenções: Apresenta um resumo do status dos planos de manutenção, destacando tarefas pendentes, atrasadas e concluídas, oferecendo uma visão clara do andamento das atividades preventivas.
    - Visão Geral dos Dispositivos Ativos: Fornece KPIs (Indicadores Chave de Desempenho) e totalizadores sobre o inventário, como o número de dispositivos ativos versus inativos, a distribuição por setor e alertas para equipamentos que necessitam de atenção.
    - Central de Chamados: Mostra estatísticas vitais sobre o suporte técnico, como o número de chamados abertos por prioridade e o tempo médio de atendimento.

- Modelos Envolvidos: MonitoramentoHardware, Chamados, Dispositivos, PlanoManuPrevent.

8.2. Módulo de Dispositivos (Inventário)

Este é o módulo central para a gestão de ativos de TI. Ele permite o controle completo do ciclo de vida de todos os equipamentos de hardware do hospital.

- Funções Principais:
    - CRUD (Criar, Ler, Atualizar, Excluir) para todos os tipos de dispositivos.
    - Visualização detalhada das especificações de hardware, software instalado e informações de rede de cada ativo.
    - Associação de dispositivos a usuários e setores, permitindo rastreabilidade.
    - Histórico completo de chamados e manutenções para cada dispositivo.
- Modelos Envolvidos: Dispositivos, Computadores, Servidores, Impressoras, Roteadores.

8.3. Módulo de Planos de Manutenção

Focado na gestão proativa de TI, este módulo permite o agendamento e acompanhamento de manutenções preventivas, essenciais para garantir a longevidade e o bom funcionamento dos equipamentos críticos.

- Funções Principais:
    - Criação de planos de manutenção com descrição, data e dispositivo associado.
    - Acompanhamento do status de cada tarefa de manutenção (Feito, Não Feito, Atrasado).
    - Geração de um cronograma de atividades para a equipe técnica.
- Modelos Envolvidos: PlanoManuPrevent, Dispositivos.

8.4. Módulo de Relatórios

Este módulo é a ferramenta de inteligência do sistema, transformando dados brutos em informações estratégicas para a tomada de decisão e avaliação da qualidade de serviço do setor.

- Funções Principais:

    - Geração de Relatórios: Permite ao usuário selecionar diferentes tipos de relatórios (ex: Inventário por Setor, chamados por Período, Custos de Manutenção).
    - Processamento de Dados: No backend, a biblioteca panda é utilizada para realizar agregações e cálculos complexos sobre os dados solicitados.
    - Exportação em PDF: Os relatórios gerados são renderizados em um template HTML e convertidos para o formato PDF com a biblioteca xhtml2pdf, permitindo o download e o compartilhamento.
- Modelos Envolvidos: Praticamente todos, dependendo do relatório gerado.

8.5. Módulo de Treinamentos

Centraliza o gerenciamento de capacitações e treinamentos realizados pela equipe de TI ou para outros setores do hospital.

- Funções Principais:
    - Cadastro de treinamentos com título, descrição, datas e setor alvo.
    - Registro de participantes e controle de presença.
    - Manutenção de um histórico de qualificações da equipe.
- Modelos Envolvidos: Treinamentos, Setores, Usuarios.

8.6. Módulo de Chamados (Sistema de Ticketing)

É o módulo operacional para o suporte técnico. Ele gerencia todo o fluxo de solicitações, desde a abertura pelo usuário até a resolução pelo técnico.

- Funções Principais:
    - Abertura de novos chamados com descrição, prioridade e categorização.
    - Atribuição de chamados a técnicos específicos.
    - Acompanhamento do status do chamado em tempo real.
    - Registro de todas as interações e da solução aplicada.
- Modelos Envolvidos: Chamados, CategoriasAtend, Dispositivos, Usuarios.

8.7. Módulo "Mais" (Utilitários e Credenciais)

Este menu agrupa funcionalidades auxiliares e repositórios de informações que, embora não façam parte dos fluxos principais, são de consulta frequente pela equipe de TI.

- Funções Principais:
    - Logins de PC, Pastas Públicas, Emails: Atuam como um cofre de credenciais ou um banco de dados de consulta rápida para informações de acesso a recursos compartilhados.
    - Hosts: Gerencia uma lista de dispositivos de rede importantes e seus respectivos endereços IP.
- Modelos Envolvidos: LoginUsuarioPc, PastaPublica, Hosts, EmailsNovos, EmailsAntigos.

8.8. Módulo de Perfil e Autenticação

Representado pelo menu de usuário no canto superior direito, este módulo lida com a sessão do usuário logado.

- Funções Principais:
    - Autenticação: Controla o processo de login e logout do sistema.
    - Gerenciamento de Perfil: Permite ao usuário visualizar seus dados e alterar sua senha.
    - Controle de Permissões: O sistema exibe os módulos e funcionalidades de acordo com o nível de acesso do usuário (no exemplo, "Analista de Dados").
- Modelos Envolvidos: AuthUser, AuthGroup, AuthPermission.

9. Módulo de Relatórios e Cálculos Matemáticos

Este módulo é o cérebro analítico do sistema, responsável por duas funções distintas, porém complementares: a geração de relatórios formais para documentação e a execução de cálculos complexos para fornecer métricas de inteligência de negócios, como a saúde dos ativos e a performance do setor de TI.

9.1. Emissão de Relatórios (PDF)

O sistema permite a geração de relatórios estáticos e formatados, ideais para apresentações, auditorias ou registros formais. O processo ocorre da seguinte forma:

1. Interface do Usuário: O usuário seleciona o tipo de relatório desejado (ex: "Inventário Geral", "Chamados por Período") e define os filtros necessários (datas, setores etc.).
2. Coleta e Agregação de Dados: A view no backend Django recebe a solicitação e consulta o banco de dados através do ORM. Para relatórios que exigem cálculos complexos (médias, somatórios, agrupamentos), a biblioteca panda é utilizada para manipular os dados em memória, transformando-os em tabelas e sumários.
3. Renderização: Os dados processados são injetados em um template HTML, que estrutura visualmente o relatório com tabelas, títulos e, se necessário, imagens de gráficos.
4. Conversão para PDF: A biblioteca xhtml2pdf processa o HTML renderizado e o converte em um documento PDF, que é então disponibilizado para download pelo usuário.

OBS: Cálculos de Depreciação ainda estão em andamento, junto ao algoritmo de machine learn

10. Software Agente de Monitoramento

O agente_gui.py é uma aplicação desktop desenvolvida em **CustomTkinter** com as seguintes responsabilidades:

1. **Instalação Local:** É instalado na máquina de cada usuário a ser monitorado.
2. **Coleta de Dados:** Periodicamente, o agente coleta informações de hardware do sistema operacional, como uso de CPU, percentual de memória RAM utilizada, espaço em disco disponível, etc.
3. **Comunicação com o Servidor:** Utilizando a biblioteca requests (ou similar), o agente envia os dados coletados para a API do servidor Django.
4. **Interface Gráfica:** Oferece uma pequena janela para que o usuário veja que o monitoramento está ativo e, possivelmente, para iniciar ou parar o serviço manualmente.

11. Conclusão

O Sistema de Gerência de Dispositivos transcende a função de uma simples ferramenta de inventário, estabelecendo-se como uma plataforma estratégica para a modernização da gestão de TI em um ambiente de missão crítica como o hospitalar. A arquitetura sólida, baseada em tecnologias confiáveis como Django e PostgreSQL, não apenas resolve os desafios atuais de controle de ativos, chamados e manutenções, mas, principalmente, constrói a fundação de dados necessária para a próxima geração de gerenciamento de TI: a inteligência preditiva.

O projeto já demonstra um valor imenso ao centralizar informações e introduzir métricas objetivas através de seus módulos de cálculo de "Índice de Saúde" e "Qualidade de Serviço". No entanto, o verdadeiro potencial do sistema reside em sua trajetória futura, que visa evoluir de uma ferramenta de gerenciamento para uma plataforma de análise avançada, impulsionada por algoritmos de Machine Learning.

A ênfase para o futuro desenvolvimento se concentra em três pilares principais:

1. Manutenção Preditiva: Os dados de monitoramento de hardware, combinados com o histórico de chamados e a idade dos equipamentos, formam o conjunto de dados ideal para treinar modelos de classificação e regressão. O objetivo é que o sistema aprenda a identificar padrões sutis que antecedem falhas de hardware, permitindo que a equipe de TI atue de forma proativa, substituindo componentes antes que eles quebrem e minimizando o tempo de inatividade de sistemas vitais.
2. Taxa de Depreciação Avançada: O modelo atual, baseado em regras, será aprimorado por algoritmos que analisarão a performance real e a frequência de falhas de cada modelo de dispositivo ao longo do tempo. Isso permitirá a criação de uma curva de depreciação data-driven, muito mais precisa, otimizando o planejamento de orçamento e o ciclo de vida dos ativos.
3. Qualidade de Serviço (QoS) Inteligente: Os modelos de Machine Learning poderão analisar o volume e o conteúdo dos chamados para identificar a causa raiz de problemas recorrentes de forma automática, prever picos de demanda de suporte e otimizar a alocação de recursos da equipe de TI.

Em suma, o Sistema de Gerência de Dispositivos é mais do que uma solução para organizar o presente; é um projeto visionário que prepara o terreno para prever e moldar o futuro das operações tecnológicas do hospital. Ao transformar dados em insights e insights em ações preditivas, a ferramenta se posiciona como um ativo de alto valor estratégico, fundamental para garantir a eficiência, a segurança e a continuidade operacional que o ambiente de saúde exige.