# Documentação do Projeto Control360 SOU

## 1. Visão Geral do Projeto

O Control360 SOU é uma aplicação web desenvolvida em Flask que fornece uma plataforma integrada para análise e gestão de dados em três níveis: Gerencial, Macro e Micro. A aplicação é estruturada em módulos independentes que trabalham em conjunto para fornecer uma visão completa dos dados e métricas do negócio.

### 1.1 Tecnologias Principais

- Python 3.x

- Flask (Framework Web)

- SQLAlchemy (ORM)

- Pandas (Manipulação de Dados)

- NumPy (Processamento Numérico)

- Bootstrap (Frontend)

## 2. Arquitetura do Sistema

### 2.1 Estrutura de Diretórios

```

app/

├── gerencial/ # Módulo de gestão gerencial

├── macro/ # Módulo de análise macro

├── micro/ # Módulo de análise micro

├── utils/ # Utilitários compartilhados

├── \_\_init\_\_.py # Configuração principal

└── error\_handlers.py # Tratamento de erros

```

### 2.2 Componentes Principais

- \*\*Blueprints\*\*: Organização modular da aplicação

- \*\*Services\*\*: Lógica de negócio

- \*\*Routes\*\*: Endpoints da API

- \*\*Templates\*\*: Interface do usuário

- \*\*Static\*\*: Recursos estáticos (CSS, JS, imagens)

## 3. Módulos Principais

### 3.1 Módulo Gerencial

O módulo gerencial é responsável pela visão estratégica do negócio, fornecendo dashboards e análises de alto nível.

#### 3.1.1 Principais Funcionalidades

- Dashboard gerencial

- Análise de performance

- Relatórios executivos

- KPIs estratégicos

#### 3.1.2 Arquivos Principais

- `routes.py`: Definição das rotas

- `services.py`: Lógica de negócio

- `base\_service.py`: Classes base

- `constants.py`: Constantes e configurações

### 3.2 Módulo Macro

O módulo macro foca em análises de médio prazo e tendências do mercado.

#### 3.2.1 Principais Funcionalidades

- Análise de mercado

- Tendências e previsões

- Comparativos setoriais

- Indicadores macroeconômicos

### 3.3 Módulo Micro

O módulo micro concentra-se em análises detalhadas e operacionais.

#### 3.3.1 Principais Funcionalidades

- Análise operacional

- Métricas de performance

- Indicadores de eficiência

- Relatórios detalhados

## 4. KPIs e Métricas

### 4.1 KPIs Gerenciais

- Taxa de crescimento

- Margem de lucro

- ROI

- Market share

### 4.2 KPIs Macro

- Crescimento do mercado

- Participação de mercado

- Tendências setoriais

- Indicadores econômicos

### 4.3 KPIs Micro

- Eficiência operacional

- Produtividade

- Qualidade

- Tempo de ciclo

## 5. Rotas e Endpoints

### 5.1 Rotas Gerenciais

#### 5.1.1 Dashboard Principal

- \*\*Rota\*\*: `/gerencial/`

- \*\*Método\*\*: GET

- \*\*Parâmetros\*\*:

- `squad`: Filtro por squad (opcional)

- `faturamento`: Filtro por tipo de faturamento (opcional)

- \*\*Funcionalidades\*\*:

- Exibe métricas gerais

- Gráficos de projetos por squad

- Gráficos de projetos por faturamento

- Lista de projetos críticos

- Ocupação das squads

#### 5.1.2 APIs

- \*\*Projetos Ativos\*\*

- Rota: `/gerencial/api/projetos-ativos`

- Retorna lista de projetos em andamento

- \*\*Projetos Críticos\*\*

- Rota: `/gerencial/api/projetos-criticos`

- Retorna projetos com status crítico

- Inclui métricas de bloqueados, horas negativas e atrasados

- \*\*Projetos em Atendimento\*\*

- Rota: `/gerencial/api/projetos-em-atendimento`

- Lista projetos em fase de atendimento

- \*\*Projetos para Faturar\*\*

- Rota: `/gerencial/api/projetos-para-faturar`

- Retorna projetos pendentes de faturamento

### 5.1.3 Métricas e KPIs

- \*\*Métricas Principais\*\*:

- Total de projetos

- Projetos ativos

- Projetos abertos

- Taxa de burn rate

- \*\*Filtros Disponíveis\*\*:

- Por Squad

- Por Tipo de Faturamento

- Por Status do Projeto

### 5.2 Rotas Macro

#### 5.2.1 Dashboard Principal

- \*\*Rota\*\*: `/macro/`

- \*\*Método\*\*: GET

- \*\*Funcionalidades\*\*:

- KPIs macro

- Projetos ativos

- Projetos críticos

- Projetos concluídos

- Eficiência de entrega

- Projetos em risco

- Tempo médio de vida

- Ocupação de squads

#### 5.2.2 APIs

- \*\*Especialistas\*\*

- Rota: `/macro/api/especialistas`

- Retorna dados de alocação dos especialistas

- Inclui taxa de uso e projetos ativos

- \*\*Accounts\*\*

- Rota: `/macro/api/accounts`

- Retorna dados de accounts e seus projetos

- Inclui métricas por account manager

- \*\*Filtros\*\*

- Rota: `/macro/api/filter`

- Parâmetros:

- `squad`: Filtro por squad

- `especialista`: Filtro por especialista

- `date`: Filtro por data

- Retorna dados filtrados com KPIs e agregações

- \*\*Projetos por Especialista\*\*

- Rota: `/macro/api/projetos/especialista/<nome\_especialista>`

- Retorna projetos ativos do especialista

- Inclui status, horas restantes e conclusão

- \*\*Projetos por Account\*\*

- Rota: `/macro/api/projetos/account/<nome\_account>`

- Retorna projetos associados à account

- Inclui métricas e status

#### 5.2.3 Métricas e KPIs

- \*\*Métricas Principais\*\*:

- Projetos ativos

- Projetos críticos

- Média de horas

- Projetos concluídos

- Eficiência de entrega

- Tempo médio de vida

- \*\*Indicadores de Status\*\*:

- Novo (azul claro)

- Em atendimento (azul)

- Aguardando (amarelo)

- Encerrado/Resolvido/Fechado (verde)

- Bloqueado (preto)

- Atrasado (amarelo)

- Cancelado (vermelho)

- \*\*Indicadores de Conclusão\*\*:

- ≥90% (verde)

- ≥70% (azul)

- ≥50% (amarelo)

- <50% (vermelho)

### 5.3 Rotas Micro

#### 5.3.1 Dashboard Principal

- \*\*Rota\*\*: `/micro/`

- \*\*Método\*\*: GET

- \*\*Funcionalidades\*\*:

- Métricas micro

- Visualização detalhada de projetos

- Análise de eficiência

- Indicadores operacionais

#### 5.3.2 APIs

- \*\*Projetos por Especialista\*\*

- Rota: `/micro/api/projetos/especialista/<nome\_especialista>`

- Retorna projetos detalhados do especialista

- Inclui métricas e status

- \*\*Projetos por Account\*\*

- Rota: `/micro/api/projetos/account/<nome\_account>`

- Retorna projetos detalhados da account

- Inclui métricas e status

- \*\*Projetos Ativos\*\*

- Rota: `/micro/api/projetos/ativos`

- Retorna lista de projetos em andamento

- Inclui detalhes operacionais

- \*\*Projetos Críticos\*\*

- Rota: `/micro/api/projetos/criticos`

- Retorna projetos em situação crítica

- Inclui indicadores de risco

- \*\*Projetos Concluídos\*\*

- Rota: `/micro/api/projetos/concluidos`

- Retorna projetos finalizados

- Inclui métricas de conclusão

- \*\*Projetos por Eficiência\*\*

- Rota: `/micro/api/projetos/eficiencia`

- Retorna projetos ordenados por eficiência

- Inclui indicadores de performance

#### 5.3.3 Métricas e KPIs

- \*\*Métricas Operacionais\*\*:

- Tempo de ciclo

- Taxa de conclusão

- Eficiência operacional

- Qualidade de entrega

- \*\*Indicadores de Performance\*\*:

- Velocidade de entrega

- Taxa de retrabalho

- Satisfação do cliente

- Aderência ao planejamento

## 6. Serviços e Funções

### 6.1 Serviços Gerenciais

#### 6.1.1 GerencialService

- \*\*Carregamento de Dados\*\*

- `carregar\_dados()`: Carrega e processa dados do CSV

- `converter\_tempo\_para\_horas()`: Converte formatos de tempo para horas decimais

- `calcular\_horas\_restantes()`: Calcula horas restantes dos projetos

- \*\*Processamento de Projetos\*\*

- `obter\_projetos\_ativos()`: Lista projetos em andamento

- `obter\_projetos\_criticos()`: Identifica projetos em situação crítica

- `obter\_projetos\_em\_atendimento()`: Lista projetos em atendimento

- `obter\_projetos\_para\_faturar()`: Identifica projetos pendentes de faturamento

- \*\*Cálculos e Métricas\*\*

- `processar\_gerencial()`: Processa dados para o dashboard gerencial

- `calcular\_metricas\_avancadas()`: Calcula KPIs e métricas complexas

- `calcular\_alertas()`: Identifica situações que requerem atenção

- `obter\_metricas\_gerencial()`: Retorna métricas consolidadas

#### 6.1.2 Constantes e Configurações

- \*\*Status de Projetos\*\*

- Status não ativos: FECHADO, ENCERRADO, RESOLVIDO, CANCELADO

- Status em andamento: NOVO, AGUARDANDO, BLOQUEADO, EM ATENDIMENTO

- Status atrasado: ATRASADO

- Status ativo: ATIVO

- \*\*Tipos de Faturamento\*\*

- PRIME: Projetos premium

- PLUS: Projetos com desconto inicial

- INICIO: Faturamento no início

- TERMINO: Faturamento no término

- FEOP: Faturado em outro projeto

- ENGAJAMENTO: Projetos de engajamento

### 6.2 Serviços Macro

#### 6.2.1 MacroService

- \*\*Análise de Mercado\*\*

- `calcular\_projetos\_ativos()`: Análise de projetos em andamento

- `calcular\_projetos\_criticos()`: Identificação de projetos críticos

- `calcular\_media\_horas()`: Análise de esforço médio

- `calcular\_projetos\_concluidos()`: Análise de conclusões

- \*\*Métricas de Performance\*\*

- `calcular\_eficiencia\_entrega()`: Análise de eficiência

- `calcular\_projetos\_risco()`: Identificação de riscos

- `calcular\_tempo\_medio\_vida()`: Análise de ciclo de vida

- `calcular\_alocacao\_especialistas()`: Análise de alocação

### 6.3 Serviços Micro

#### 6.3.1 MicroService

- \*\*Análise Operacional\*\*

- `obter\_metricas\_micro()`: Cálculo de métricas operacionais

- `obter\_projetos\_por\_especialista()`: Análise por especialista

- `obter\_projetos\_por\_account()`: Análise por account

- `obter\_projetos\_eficiencia()`: Análise de eficiência

- \*\*Métricas Detalhadas\*\*

- Tempo de ciclo por projeto

- Taxa de conclusão por especialista

- Eficiência operacional por squad

- Qualidade de entrega por account

## 7. Configurações e Dependências

### 7.1 Requisitos do Sistema

```

Flask==2.0.1

SQLAlchemy==1.4.23

pandas==1.3.3

numpy==1.21.2

python-dateutil==2.8.2

```

### 7.2 Configurações

#### 7.2.1 Ambiente de Desenvolvimento

- Python 3.x

- Virtualenv para isolamento

- Debug mode ativado

- Logging detalhado

#### 7.2.2 Ambiente de Produção

- Gunicorn como servidor WSGI

- Logging em arquivo com rotação

- Cache configurado

- Segurança e autenticação

#### 7.2.3 Logs e Monitoramento

- Sistema de logging configurado

- Rotação de arquivos de log

- Níveis de log configuráveis

- Monitoramento de erros

## 8. Manutenção e Suporte

### 8.1 Logs

- Sistema de logging configurado

- Rotação de arquivos de log

- Níveis de log configuráveis

- Monitoramento de erros

### 8.2 Backup e Recuperação

- Estratégias de backup

- Procedimentos de recuperação

- Manutenção de dados

- Segurança da informação

## 9. Estrutura de Dados e Modelos

### 9.1 Modelo de Dados

#### 9.1.1 Projeto

- \*\*Campos Principais\*\*

- `Numero`: Identificador único do projeto

- `Projeto`: Nome do projeto/cliente

- `Status`: Estado atual do projeto

- `Squad`: Equipe responsável

- `Faturamento`: Tipo de faturamento

- `Horas`: Esforço estimado

- `HorasTrabalhadas`: Tempo já dedicado

- `HorasRestantes`: Saldo de horas

- `Conclusao`: Percentual concluído

- `Especialista`: Responsável técnico

- `Account Manager`: Gerente de conta

- \*\*Campos de Data\*\*

- `DataInicio`: Data de abertura

- `DataTermino`: Data de conclusão

- `VencimentoEm`: Data prevista de término

- `UltimaInteracao`: Última atualização

#### 9.1.2 Especialista

- \*\*Campos Principais\*\*

- `Nome`: Nome do especialista

- `Squad`: Equipe de alocação

- `ProjetosAtivos`: Lista de projetos

- `TaxaUso`: Percentual de ocupação

- `HorasDisponiveis`: Capacidade restante

#### 9.1.3 Account

- \*\*Campos Principais\*\*

- `Nome`: Nome do account manager

- `Projetos`: Lista de projetos

- `FaturamentoTotal`: Valor faturado

- `ProjetosAtivos`: Quantidade atual

- `SatisfacaoCliente`: Índice de satisfação

### 9.2 Relacionamentos

- \*\*Projeto -> Especialista\*\*: Muitos para um

- \*\*Projeto -> Account\*\*: Muitos para um

- \*\*Projeto -> Squad\*\*: Muitos para um

- \*\*Especialista -> Squad\*\*: Muitos para um

### 9.3 Validações e Regras

#### 9.3.1 Validações de Dados

- Status deve ser um dos valores permitidos

- Datas devem estar em formato válido

- Horas não podem ser negativas

- Conclusão deve estar entre 0 e 100%

#### 9.3.2 Regras de Negócio

- Projetos críticos: horas negativas ou atraso > 7 dias

- Taxa de uso: soma das horas trabalhadas / capacidade total

- Eficiência: horas estimadas / horas realizadas

- Satisfação: média das avaliações dos últimos 3 meses

### 9.4 Transformações de Dados

#### 9.4.1 Conversões

- Tempo: HH:MM:SS -> Horas decimais

- Percentuais: String -> Float

- Datas: DD/MM/YYYY -> DateTime

- Status: Texto -> Código padronizado

#### 9.4.2 Agregações

- Total de projetos por squad

- Média de horas por especialista

- Soma de faturamento por account

- Distribuição de status

## 10. Interface do Usuário e Templates

### 10.1 Estrutura de Templates

#### 10.1.1 Layout Base

- \*\*base.html\*\*

- Cabeçalho com navegação

- Menu lateral com módulos

- Área de conteúdo principal

- Rodapé com informações

#### 10.1.2 Módulo Gerencial

- \*\*dashboard.html\*\*

- Cards de KPIs principais

- Gráficos de projetos por squad

- Gráficos de projetos por faturamento

- Lista de projetos críticos

- Filtros de visualização

#### 10.1.3 Módulo Macro

- \*\*dashboard.html\*\*

- Indicadores macro

- Gráficos de tendências

- Análise de mercado

- Distribuição de projetos

- Filtros avançados

#### 10.1.4 Módulo Micro

- \*\*dashboard.html\*\*

- Métricas operacionais

- Detalhes de projetos

- Análise de eficiência

- Indicadores de qualidade

- Filtros específicos

### 10.2 Componentes Reutilizáveis

#### 10.2.1 Partials

- \*\*tabela\_projetos.html\*\*

- Lista de projetos

- Ordenação e filtros

- Paginação

- Ações em lote

- \*\*grafico\_status.html\*\*

- Gráfico de status

- Legenda interativa

- Tooltips informativos

- Exportação de dados

- \*\*filtros\_avancados.html\*\*

- Filtros por data

- Filtros por squad

- Filtros por especialista

- Filtros por account

#### 10.2.2 Widgets

- \*\*card\_kpi.html\*\*

- Valor principal

- Variação percentual

- Ícone indicativo

- Link para detalhes

- \*\*grafico\_linha.html\*\*

- Série temporal

- Múltiplas séries

- Zoom e pan

- Exportação

- \*\*tabela\_especialistas.html\*\*

- Lista de especialistas

- Alocação atual

- Projetos ativos

- Indicadores de performance

### 10.3 Estilos e Temas

#### 10.3.1 Cores e Status

- \*\*Status de Projetos\*\*

- Novo: Azul claro (#17a2b8)

- Em atendimento: Azul (#007bff)

- Aguardando: Amarelo (#ffc107)

- Concluído: Verde (#28a745)

- Bloqueado: Preto (#343a40)

- Atrasado: Amarelo (#ffc107)

- Cancelado: Vermelho (#dc3545)

- \*\*Indicadores de Progresso\*\*

- Alto: Verde (#28a745)

- Médio: Azul (#17a2b8)

- Baixo: Amarelo (#ffc107)

- Crítico: Vermelho (#dc3545)

#### 10.3.2 Componentes

- \*\*Cards\*\*

- Sombra suave

- Bordas arredondadas

- Hover effects

- Animações suaves

- \*\*Tabelas\*\*

- Linhas alternadas

- Hover em linhas

- Ordenação por colunas

- Responsividade

- \*\*Gráficos\*\*

- Cores consistentes

- Tooltips informativos

- Legendas interativas

- Exportação em PNG/CSV

### 10.4 Responsividade

- \*\*Breakpoints\*\*

- Mobile: < 576px

- Tablet: 576px - 768px

- Desktop: > 768px

- \*\*Adaptações\*\*

- Menu colapsável em mobile

- Cards em grid responsivo

- Tabelas com scroll horizontal

- Gráficos redimensionáveis

## 11. Segurança e Autenticação

### 11.1 Autenticação

#### 11.1.1 Métodos de Autenticação

- \*\*Login com Credenciais\*\*

- Usuário e senha

- Senha com requisitos mínimos

- Bloqueio após tentativas falhas

- Recuperação de senha

- \*\*Autenticação em Dois Fatores (2FA)\*\*

- Código por SMS

- Código por email

- Aplicativo autenticador

- Backup codes

#### 11.1.2 Níveis de Acesso

- \*\*Administrador\*\*

- Acesso total ao sistema

- Gestão de usuários

- Configurações do sistema

- Logs e auditoria

- \*\*Gerente\*\*

- Acesso a todos os módulos

- Relatórios gerenciais

- Gestão de projetos

- Análises avançadas

- \*\*Especialista\*\*

- Acesso aos seus projetos

- Atualização de status

- Registro de horas

- Relatórios básicos

- \*\*Account Manager\*\*

- Acesso aos seus clientes

- Gestão de projetos

- Relatórios de faturamento

- Indicadores de satisfação

### 11.2 Segurança de Dados

#### 11.2.1 Proteção de Dados

- \*\*Criptografia\*\*

- Dados em trânsito (HTTPS)

- Dados em repouso

- Senhas com hash seguro

- Chaves de API criptografadas

- \*\*Backup e Recuperação\*\*

- Backup automático diário

- Retenção de 30 dias

- Backup incremental

- Teste de recuperação

#### 11.2.2 Controle de Acesso

- \*\*Políticas de Acesso\*\*

- RBAC (Role-Based Access Control)

- Permissões granulares

- Auditoria de acessos

- Registro de atividades

- \*\*Sessões\*\*

- Timeout automático

- Sessão única

- Revogação de tokens

- Logout automático

### 11.3 Logs e Auditoria

#### 11.3.1 Registro de Atividades

- \*\*Logs de Sistema\*\*

- Acessos ao sistema

- Alterações de dados

- Erros e exceções

- Performance

- \*\*Logs de Negócio\*\*

- Criação de projetos

- Alterações de status

- Atualizações de horas

- Faturamentos

#### 11.3.2 Monitoramento

- \*\*Alertas\*\*

- Tentativas de acesso suspeitas

- Erros críticos

- Performance degradada

- Uso anormal de recursos

- \*\*Relatórios\*\*

- Relatório de acessos

- Relatório de atividades

- Relatório de erros

- Relatório de segurança

### 11.4 Conformidade

#### 11.4.1 LGPD

- \*\*Tratamento de Dados\*\*

- Consentimento do usuário

- Finalidade específica

- Tempo de retenção

- Direitos do titular

- \*\*Medidas de Segurança\*\*

- Controle de acesso

- Registro de atividades

- Plano de resposta a incidentes

- Avaliação de riscos

#### 11.4.2 Políticas

- \*\*Política de Privacidade\*\*

- Coleta de dados

- Uso de informações

- Compartilhamento

- Direitos do usuário

- \*\*Política de Segurança\*\*

- Requisitos de senha

- Uso aceitável

- Incidentes de segurança

- Procedimentos de emergência

## 12. Testes e Qualidade

### 12.1 Testes Unitários

#### 12.1.1 Serviços

- \*\*GerencialService\*\*

- Teste de carregamento de dados

- Teste de cálculos de métricas

- Teste de validações

- Teste de transformações

- \*\*MacroService\*\*

- Teste de análises de mercado

- Teste de cálculos de eficiência

- Teste de projeções

- Teste de agregações

- \*\*MicroService\*\*

- Teste de métricas operacionais

- Teste de cálculos de performance

- Teste de indicadores

- Teste de relatórios

#### 12.1.2 Utilitários

- \*\*Conversores\*\*

- Teste de conversão de tempo

- Teste de formatação de datas

- Teste de cálculos numéricos

- Teste de validações

- \*\*Validadores\*\*

- Teste de regras de negócio

- Teste de formatos de dados

- Teste de consistência

- Teste de integridade

### 12.2 Testes de Integração

#### 12.2.1 APIs

- \*\*Endpoints\*\*

- Teste de rotas

- Teste de parâmetros

- Teste de respostas

- Teste de erros

- \*\*Fluxos\*\*

- Teste de processos completos

- Teste de dependências

- Teste de transações

- Teste de concorrência

#### 12.2.2 Banco de Dados

- \*\*Operações\*\*

- Teste de inserções

- Teste de atualizações

- Teste de consultas

- Teste de exclusões

- \*\*Integridade\*\*

- Teste de constraints

- Teste de relacionamentos

- Teste de índices

- Teste de performance

### 12.3 Testes de Interface

#### 12.3.1 Componentes

- \*\*Templates\*\*

- Teste de renderização

- Teste de dados

- Teste de interações

- Teste de responsividade

- \*\*JavaScript\*\*

- Teste de funções

- Teste de eventos

- Teste de AJAX

- Teste de validações

#### 12.3.2 Usabilidade

- \*\*Interface\*\*

- Teste de navegação

- Teste de acessibilidade

- Teste de responsividade

- Teste de performance

- \*\*Experiência\*\*

- Teste de usabilidade

- Teste de feedback

- Teste de erros

- Teste de ajuda

### 12.4 Qualidade de Código

#### 12.4.1 Análise Estática

- \*\*Linting\*\*

- PEP 8

- ESLint

- Style Guide

- Best Practices

- \*\*Complexidade\*\*

- Cíclomática

- Halstead

- Manutenibilidade

- Legibilidade

#### 12.4.2 Métricas

- \*\*Cobertura\*\*

- Cobertura de código

- Cobertura de branches

- Cobertura de funções

- Cobertura de linhas

- \*\*Performance\*\*

- Tempo de resposta

- Uso de memória

- Uso de CPU

- Tempo de carregamento

### 12.5 Automação

#### 12.5.1 CI/CD

- \*\*Integração Contínua\*\*

- Build automático

- Testes automáticos

- Análise de código

- Notificações

- \*\*Entrega Contínua\*\*

- Deploy automático

- Rollback automático

- Versionamento

- Ambiente de staging

#### 12.5.2 Monitoramento

- \*\*Métricas\*\*

- Uptime

- Performance

- Erros

- Uso de recursos

- \*\*Alertas\*\*

- Thresholds

- Notificações

- Escalação

- Resolução

## 13. Implantação e Infraestrutura

### 13.1 Ambientes

#### 13.1.1 Desenvolvimento

- \*\*Configuração\*\*

- Python 3.x

- Virtualenv

- Flask em modo debug

- SQLite para dados

- \*\*Ferramentas\*\*

- VS Code/PyCharm

- Git para versionamento

- Docker para containers

- Postman para APIs

#### 13.1.2 Homologação

- \*\*Configuração\*\*

- Python 3.x

- Gunicorn

- PostgreSQL

- Redis para cache

- \*\*Monitoramento\*\*

- Logs detalhados

- Métricas de performance

- Alertas de erro

- Backup automático

#### 13.1.3 Produção

- \*\*Configuração\*\*

- Python 3.x

- Gunicorn com workers

- PostgreSQL cluster

- Redis cluster

- Nginx como proxy

- \*\*Segurança\*\*

- SSL/TLS

- Firewall

- WAF

- DDoS protection

### 13.2 Infraestrutura

#### 13.2.1 Servidores

- \*\*Aplicação\*\*

- 2+ instâncias

- Auto-scaling

- Load balancing

- Health checks

- \*\*Banco de Dados\*\*

- Master-Slave

- Replicação

- Backup automático

- Point-in-time recovery

#### 13.2.2 Rede

- \*\*DNS\*\*

- Domínio principal

- Subdomínios

- SSL wildcard

- CDN

- \*\*Segurança\*\*

- VPC

- Security Groups

- NACL

- VPN

### 13.3 Deploy

#### 13.3.1 Processo

- \*\*Preparação\*\*

- Versionamento

- Changelog

- Testes automatizados

- Backup

- \*\*Execução\*\*

- Deploy blue-green

- Rollback automático

- Health checks

- Monitoramento

#### 13.3.2 Automação

- \*\*CI/CD\*\*

- GitHub Actions

- Jenkins

- Ansible

- Terraform

- \*\*Scripts\*\*

- Deploy

- Backup

- Rollback

- Manutenção

### 13.4 Monitoramento

#### 13.4.1 Métricas

- \*\*Sistema\*\*

- CPU

- Memória

- Disco

- Rede

- \*\*Aplicação\*\*

- Requests/s

- Response time

- Error rate

- Active users

#### 13.4.2 Logs

- \*\*Aplicação\*\*

- Access logs

- Error logs

- Audit logs

- Performance logs

- \*\*Sistema\*\*

- System logs

- Security logs

- Database logs

- Network logs

### 13.5 Manutenção

#### 13.5.1 Rotina

- \*\*Backup\*\*

- Diário

- Semanal

- Mensal

- Anual

- \*\*Limpeza\*\*

- Logs antigos

- Dados temporários

- Cache

- Sessões expiradas

#### 13.5.2 Atualizações

- \*\*Sistema\*\*

- Security patches

- OS updates

- Dependencies

- Drivers

- \*\*Aplicação\*\*

- Versões

- Features

- Bug fixes

- Performance

## 14. Troubleshooting e Suporte

### 14.1 Diagnóstico de Problemas

#### 14.1.1 Logs

- \*\*Aplicação\*\*

- Logs de erro

- Stack traces

- Request/Response

- Performance

- \*\*Sistema\*\*

- Logs do sistema

- Logs de banco

- Logs de rede

- Logs de segurança

#### 14.1.2 Métricas

- \*\*Performance\*\*

- Tempo de resposta

- Uso de recursos

- Taxa de erro

- Throughput

- \*\*Recursos\*\*

- CPU

- Memória

- Disco

- Rede

### 14.2 Problemas Comuns

#### 14.2.1 Aplicação

- \*\*Erros 500\*\*

- Verificar logs

- Verificar banco

- Verificar cache

- Verificar memória

- \*\*Lentidão\*\*

- Verificar queries

- Verificar cache

- Verificar recursos

- Verificar rede

#### 14.2.2 Dados

- \*\*Inconsistências\*\*

- Verificar integridade

- Verificar constraints

- Verificar triggers

- Verificar índices

- \*\*Sincronização\*\*

- Verificar replicação

- Verificar jobs

- Verificar filas

- Verificar cache

### 14.3 Procedimentos

#### 14.3.1 Emergência

- \*\*Incidente\*\*

- Identificação

- Escalação

- Resolução

- Post-mortem

- \*\*Recuperação\*\*

- Backup

- Rollback

- Failover

- Restauração

#### 14.3.2 Manutenção

- \*\*Planejada\*\*

- Agendamento

- Notificação

- Execução

- Validação

- \*\*Não Planejada\*\*

- Detecção

- Análise

- Correção

- Documentação

### 14.4 Suporte

#### 14.4.1 Níveis

- \*\*N1 - Suporte Básico\*\*

- Problemas simples

- Dúvidas básicas

- Acesso inicial

- Documentação

- \*\*N2 - Suporte Técnico\*\*

- Problemas complexos

- Análise técnica

- Configurações

- Integrações

- \*\*N3 - Suporte Especializado\*\*

- Problemas críticos

- Desenvolvimento

- Arquitetura

- Segurança

#### 14.4.2 Canais

- \*\*Comunicação\*\*

- Email

- Chat

- Telefone

- Ticket

- \*\*Documentação\*\*

- Base de conhecimento

- FAQs

- Tutoriais

- Vídeos

### 14.5 Recuperação

#### 14.5.1 Backup

- \*\*Tipos\*\*

- Completo

- Incremental

- Diferencial

- Log

- \*\*Retenção\*\*

- Diário: 7 dias

- Semanal: 4 semanas

- Mensal: 12 meses

- Anual: 5 anos

#### 14.5.2 DR (Disaster Recovery)

- \*\*Plano\*\*

- RTO (Recovery Time Objective)

- RPO (Recovery Point Objective)

- Procedimentos

- Testes

- \*\*Ambiente\*\*

- Hot site

- Warm site

- Cold site

- Cloud

## 15. Evolução e Roadmap

### 15.1 Versões

#### 15.1.1 Histórico

- \*\*v1.0.0 (2024-03)\*\*

- Lançamento inicial

- Módulos básicos

- Funcionalidades core

- Interface principal

- \*\*v1.1.0 (2024-04)\*\*

- Melhorias de performance

- Correções de bugs

- Novos relatórios

- Ajustes de UI

#### 15.1.2 Próximas Versões

- \*\*v1.2.0 (2024-05)\*\*

- Novos KPIs

- Dashboards avançados

- Integrações adicionais

- Melhorias de UX

- \*\*v2.0.0 (2024-Q3)\*\*

- Refatoração major

- Nova arquitetura

- Novos módulos

- API v2

### 15.2 Roadmap

#### 15.2.1 Curto Prazo (3 meses)

- \*\*Performance\*\*

- Otimização de queries

- Melhorias de cache

- Redução de latência

- Escalabilidade

- \*\*Funcionalidades\*\*

- Novos relatórios

- Filtros avançados

- Exportação de dados

- Notificações

#### 15.2.2 Médio Prazo (6 meses)

- \*\*Arquitetura\*\*

- Microserviços

- Event-driven

- CQRS

- Cache distribuído

- \*\*Integrações\*\*

- Sistemas externos

- APIs públicas

- Webhooks

- SSO

### 15.3 Melhorias

#### 15.3.1 Técnicas

- \*\*Código\*\*

- Refatoração

- Testes

- Documentação

- Padrões

- \*\*Infraestrutura\*\*

- Cloud native

- Containers

- Serverless

- DevOps

#### 15.3.2 Negócio

- \*\*Processos\*\*

- Automação

- Workflows

- Regras

- Validações

- \*\*Análises\*\*

- BI

- ML

- AI

- Predições

### 15.4 Pesquisa

#### 15.4.1 Tecnologias

- \*\*Frontend\*\*

- React/Vue

- TypeScript

- GraphQL

- WebAssembly

- \*\*Backend\*\*

- FastAPI

- gRPC

- Kafka

- Elasticsearch

#### 15.4.2 Tendências

- \*\*Arquitetura\*\*

- Event sourcing

- CQRS

- DDD

- Microservices

- \*\*DevOps\*\*

- GitOps

- Platform engineering

- SRE

- Observability

### 15.5 Feedback

#### 15.5.1 Coleta

- \*\*Usuários\*\*

- Pesquisas

- Entrevistas

- Analytics

- Tickets

- \*\*Equipe\*\*

- Retrospectivas

- Reviews

- Métricas

- Sugestões

#### 15.5.2 Análise

- \*\*Dados\*\*

- Uso

- Performance

- Erros

- Satisfação

- \*\*Ações\*\*

- Priorização

- Planejamento

- Implementação

- Validação

## 16. Código Fonte

### 16.1 Estrutura de Diretórios

```

app/

├── \_\_init\_\_.py # Configuração principal da aplicação

├── gerencial/ # Módulo gerencial

│ ├── \_\_init\_\_.py

│ ├── base\_service.py

│ ├── constants.py

│ ├── routes.py

│ └── services.py

├── macro/ # Módulo macro

│ ├── \_\_init\_\_.py

│ ├── base\_service.py

│ ├── constants.py

│ ├── routes.py

│ └── services.py

├── micro/ # Módulo micro

│ ├── \_\_init\_\_.py

│ ├── base\_service.py

│ ├── constants.py

│ ├── routes.py

│ └── services.py

└── utils/ # Utilitários compartilhados

├── \_\_init\_\_.py

├── base\_service.py

├── constants.py

└── json\_provider.py

```

### 16.2 Módulo Principal (app/\_\_init\_\_.py)

```python

from flask import Flask, redirect, url\_for

import logging

from logging.handlers import RotatingFileHandler

import os

from pathlib import Path

from .utils.json\_provider import NumpyJSONProvider

BASE\_DIR = Path(\_\_file\_\_).parent.parent

class MarkdownFilter(logging.Filter):

def filter(self, record):

if hasattr(record, 'markdown'):

record.msg = f"\n---\n{record.msg}\n---\n"

return True

def create\_app():

app = Flask(\_\_name\_\_,

template\_folder=str(BASE\_DIR / 'templates'),

static\_folder=str(BASE\_DIR / 'static'))

app.json = NumpyJSONProvider(app)

log\_dir = BASE\_DIR / 'logs'

log\_dir.mkdir(parents=True, exist\_ok=True)

log\_file = log\_dir / 'app.log'

log\_level = logging.DEBUG

log\_format = logging.Formatter(

'%(asctime)s - %(name)s - %(levelname)s - %(message)s [in %(pathname)s:%(lineno)d]'

)

file\_handler = RotatingFileHandler(

log\_file, maxBytes=1024\*1024\*5, backupCount=3, encoding='utf-8'

)

file\_handler.setFormatter(log\_format)

file\_handler.setLevel(log\_level)

stream\_handler = logging.StreamHandler()

stream\_handler.setFormatter(log\_format)

stream\_handler.setLevel(log\_level)

app.logger.addHandler(file\_handler)

app.logger.addHandler(stream\_handler)

app.logger.setLevel(log\_level)

app.logger.addFilter(MarkdownFilter())

app.logger.info("Aplicação Flask criada e logging configurado.")

app.logger.info(f"Logs sendo escritos em: {log\_file}")

register\_blueprints(app)

@app.route('/')

def index():

return redirect(url\_for('gerencial.dashboard'))

check\_templates(app)

return app

def check\_templates(app):

app.logger.info("Verificando templates essenciais...")

required\_templates = [

'gerencial/dashboard.html',

'gerencial/erro.html',

'macro/dashboard.html',

'macro/partials/tabela\_especialistas.html',

'micro/dashboard.html',

]

all\_found = True

for template in required\_templates:

if not template\_exists(app, template):

app.logger.warning(f"⚠️ Template essencial não encontrado: {template}")

all\_found = False

if all\_found:

app.logger.info("Todos os templates essenciais verificados foram encontrados.")

```

### 16.3 Módulo Utils (app/utils/)

#### 16.3.1 Base Service (base\_service.py)

```python

import pandas as pd

from pathlib import Path

import logging

from datetime import datetime, timedelta

import os

import numpy as np

class BaseService:

"""Classe base para serviços de processamento de dados."""

def \_\_init\_\_(self):

"""Inicializa o serviço com configurações básicas."""

self.csv\_path = Path(\_\_file\_\_).parent.parent.parent / 'data' / 'dadosr.csv'

logger = logging.getLogger(\_\_name\_\_)

logger.info(f"Caminho do CSV definido para: {self.csv\_path}")

def carregar\_dados(self):

"""Carrega dados do arquivo CSV."""

try:

if not self.csv\_path.exists():

raise FileNotFoundError(f"Arquivo CSV não encontrado: {self.csv\_path}")

dados = pd.read\_csv(self.csv\_path,

sep=';',

encoding='latin1',

quoting=1) # QUOTE\_ALL

if dados.empty:

logger.warning("Arquivo CSV está vazio")

return pd.DataFrame()

# Verifica colunas obrigatórias

colunas\_obrigatorias = ['Status', 'Squad', 'Especialista', 'Conclusao']

colunas\_faltantes = [col for col in colunas\_obrigatorias if col not in dados.columns]

if colunas\_faltantes:

raise ValueError(f"Colunas obrigatórias ausentes: {colunas\_faltantes}")

# Converte datas

for col in ['DataInicio', 'DataTermino', 'VencimentoEm']:

if col in dados.columns:

dados[col] = pd.to\_datetime(dados[col], errors='coerce')

# Padroniza texto

for col in ['Status', 'Squad', 'Especialista', 'Account Manager']:

if col in dados.columns:

dados[col] = dados[col].fillna('').astype(str).str.strip()

# Converte conclusão para numérico

if 'Conclusao' in dados.columns:

dados['Conclusao'] = pd.to\_numeric(dados['Conclusao'], errors='coerce')

dados['Conclusao'] = dados['Conclusao'].clip(0, 100)

# Converte horas trabalhadas

if 'HorasTrabalhadas' in dados.columns:

dados['HorasTrabalhadas'] = dados['HorasTrabalhadas'].apply(self.converter\_tempo\_para\_horas)

# Limpa nomes de projetos

if 'Projeto' in dados.columns:

dados['Projeto'] = dados['Projeto'].apply(self.limpar\_nome\_projeto)

# Calcula horas restantes

dados = self.calcular\_horas\_restantes(dados)

logger.info(f"Dados carregados com sucesso: {len(dados)} registros")

return dados

except Exception as e:

logger.error(f"Erro ao carregar dados: {str(e)}", exc\_info=True)

return pd.DataFrame()

def calcular\_horas\_restantes(self, dados):

"""Calcula horas restantes para cada projeto."""

try:

if 'HorasTrabalhadas' not in dados.columns or 'HorasPrevistas' not in dados.columns:

return dados

dados['HorasRestantes'] = dados['HorasPrevistas'] - dados['HorasTrabalhadas']

dados['HorasRestantes'] = dados['HorasRestantes'].clip(lower=0)

return dados

except Exception as e:

logger.error(f"Erro ao calcular horas restantes: {str(e)}")

return dados

def limpar\_nome\_projeto(self, nome):

"""Remove partes do nome do projeto após vírgulas."""

try:

if pd.isna(nome) or nome == '':

return ''

nome = str(nome).strip()

if ',' in nome:

nome = nome.split(',')[0].strip()

return nome

except Exception as e:

logger.error(f"Erro ao limpar nome do projeto '{nome}': {str(e)}")

return str(nome)

```

#### 16.3.2 Constants (constants.py)

```python

# Status dos projetos

STATUS\_ATIVO = 'ATIVO'

STATUS\_CRITICO = 'CRITICO'

STATUS\_CONCLUIDO = 'CONCLUIDO'

STATUS\_ATENDIMENTO = 'EM ATENDIMENTO'

# Constantes para cálculos

DIAS\_POR\_MES = 30

HORAS\_POR\_DIA = 8

DIAS\_POR\_SEMANA = 5

# Limites para alertas

LIMITE\_HORAS\_NEGATIVAS = 0

LIMITE\_ATRASO\_DIAS = 7

LIMITE\_CONCLUSAO\_MINIMA = 30

# Cores para visualização

CORES\_STATUS = {

'NOVO': 'info',

'EM ATENDIMENTO': 'primary',

'AGUARDANDO': 'warning',

'ENCERRADO': 'success',

'RESOLVIDO': 'success',

'FECHADO': 'success',

'BLOQUEADO': 'dark',

'ATRASADO': 'warning',

'CANCELADO': 'danger',

'CRITICO': 'danger',

'CONCLUIDO': 'success'

}

# Cores para conclusão

CORES\_CONCLUSAO = {

'success': 90, # Verde

'info': 70, # Azul

'warning': 50, # Amarelo

'danger': 0 # Vermelho

}

# Colunas obrigatórias

COLUNAS\_OBRIGATORIAS = ['Projeto', 'Status', 'Squad', 'Faturamento', 'HorasTrabalhadas']

COLUNAS\_NUMERICAS = ['Horas', 'HorasRestantes', 'Conclusao', 'HorasTrabalhadas']

COLUNAS\_TEXTO = ['Squad', 'Status', 'Faturamento', 'Especialista', 'Account Manager']

```

### 16.4 Módulo Gerencial (app/gerencial/)

#### 16.4.1 Routes (routes.py)

```python

import uuid

import logging

import traceback

import pandas as pd

from flask import render\_template, request, jsonify

from . import gerencial\_bp

from .services import GerencialService

gerencial\_service = GerencialService()

logger = logging.getLogger(\_\_name\_\_)

@gerencial\_bp.route('/')

def dashboard():

"""Rota principal do dashboard gerencial"""

try:

request\_id = str(uuid.uuid4())[:8]

logger.info(f"[{request\_id}] Iniciando carregamento do dashboard gerencial.")

squad = request.args.get('squad', '').strip()

faturamento = request.args.get('faturamento', '').strip()

dados = gerencial\_service.carregar\_dados()

if dados.empty:

logger.error(f"[{request\_id}] Nenhum dado encontrado para exibição")

return render\_template('gerencial/dashboard.html',

erro="Não foi possível carregar os dados do dashboard",

codigo\_erro=request\_id,

filtro\_aplicado={'squad': '', 'faturamento': ''},

metricas={'total\_projetos': 0, 'projetos\_ativos': 0, 'projetos\_abertos': 0, 'burn\_rate': 0.0})

dados\_filtrados = dados.copy()

if squad and squad != 'Todos':

dados\_filtrados = dados\_filtrados[dados\_filtrados['Squad'] == squad]

if faturamento and faturamento != 'Todos':

dados\_filtrados = dados\_filtrados[dados\_filtrados['Faturamento'] == faturamento]

resultado = gerencial\_service.processar\_gerencial(dados\_filtrados)

return render\_template('gerencial/dashboard.html',

metricas=resultado['metricas'],

projetos\_ativos=resultado['projetos\_ativos'],

projetos\_criticos=resultado['projetos\_criticos'],

projetos\_em\_atendimento=resultado['projetos\_em\_atendimento'],

projetos\_para\_faturar=resultado['projetos\_para\_faturar'],

filtro\_aplicado={'squad': squad, 'faturamento': faturamento})

except Exception as e:

logger.error(f"[{request\_id}] Erro ao carregar dashboard: {str(e)}")

return render\_template('gerencial/erro.html',

erro="Ocorreu um erro ao carregar o dashboard",

codigo\_erro=request\_id)

```

#### 16.4.2 Services (services.py)

```python

import pandas as pd

from pathlib import Path

from datetime import datetime, timedelta

import logging

import numpy as np

import glob

import os

from .base\_service import BaseService

from .constants import \*

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger(\_\_name\_\_)

class GerencialService(BaseService):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.logger = logging.getLogger(\_\_name\_\_)

self.status\_ativos = ['NOVO', 'AGUARDANDO', 'EM ATENDIMENTO', 'BLOQUEADO']

self.status\_concluidos = ['FECHADO', 'ENCERRADO', 'RESOLVIDO', 'CANCELADO']

base\_dir = Path(\_\_file\_\_).resolve().parent.parent.parent

data\_dir = base\_dir / 'data'

self.csv\_path = data\_dir / 'dadosr.csv'

logger.info(f"Caminho do CSV definido para: {self.csv\_path}")

def obter\_projetos\_ativos(self, dados):

"""Filtra projetos ativos para o modal"""

try:

if dados is None or dados.empty:

return []

projetos = dados[dados['Status'].isin(self.status\_ativos)]

return self.\_formatar\_projetos(projetos)

except Exception as e:

logger.error(f"Erro ao obter projetos ativos: {str(e)}")

return []

def obter\_projetos\_em\_atendimento(self, dados):

"""Filtra projetos com status 'EM ATENDIMENTO' ou 'NOVO' para o modal"""

try:

dados\_limpos = dados.copy()

dados\_limpos['Status'] = dados\_limpos['Status'].str.strip().str.upper()

projetos\_filtrados = dados\_limpos[

dados\_limpos['Status'].isin(['EM ATENDIMENTO', 'NOVO'])

]

projetos\_formatados = projetos\_filtrados.copy()

if 'Conclusao' in projetos\_formatados.columns:

projetos\_formatados['Conclusao'] = projetos\_formatados['Conclusao'].fillna(0)

if 'HorasRestantes' in projetos\_formatados.columns:

projetos\_formatados['HorasRestantes'] = projetos\_formatados['HorasRestantes'].fillna(0)

if 'VencimentoEm' in projetos\_formatados.columns:

projetos\_formatados['VencimentoEm'] = pd.to\_datetime(

projetos\_formatados['VencimentoEm'],

errors='coerce'

).dt.strftime('%d/%m/%Y')

projetos\_formatados['VencimentoEm'] = projetos\_formatados['VencimentoEm'].fillna('N/A')

colunas = ['Projeto', 'Squad', 'Status', 'Conclusao', 'HorasRestantes', 'VencimentoEm']

projetos\_formatados = projetos\_formatados[colunas].copy()

return projetos\_formatados.replace({np.nan: None}).to\_dict('records')

except Exception as e:

logging.error(f"Erro ao obter projetos em atendimento: {str(e)}")

return []

def calcular\_faturamento\_pendente(self, dados):

try:

hoje = datetime.now()

mes\_atual = hoje.month

ano\_atual = hoje.year

if 'Faturamento' not in dados.columns or 'DataTermino' not in dados.columns:

return []

try:

dados['DataTermino'] = pd.to\_datetime(dados['DataTermino'], format='%d/%m/%Y', errors='coerce')

except:

try:

dados['DataTermino'] = pd.to\_datetime(dados['DataTermino'], errors='coerce')

except:

logging.error("Erro ao converter datas. Formato esperado: DD/MM/YYYY")

return []

dados = dados.dropna(subset=['DataTermino'])

cond\_inicio = dados['Faturamento'].isin(['PRIME', 'PLUS', 'INICIO'])

cond\_termino = (

dados['Faturamento'].isin(['TERMINO', 'ENGAJAMENTO']) &

(dados['DataTermino'].dt.month == mes\_atual) &

(dados['DataTermino'].dt.year == ano\_atual)

)

projetos\_faturar = dados[cond\_inicio | cond\_termino].copy()

projetos\_faturar['VencimentoEm'] = projetos\_faturar['DataTermino'].dt.strftime('%d/%m/%Y')

colunas = ['Projeto', 'Squad', 'Account Manager', 'Faturamento', 'Status', 'VencimentoEm']

projetos\_formatados = projetos\_faturar[colunas].copy()

return projetos\_formatados.replace({np.nan: None}).to\_dict('records')

except Exception as e:

logging.error(f"Erro ao calcular faturamento pendente: {str(e)}")

return []

```

#### 16.4.3 Constants (constants.py)

```python

# Grupos de status

STATUS\_NAO\_ATIVOS = ['FECHADO', 'ENCERRADO', 'RESOLVIDO', 'CANCELADO']

STATUS\_EM\_ANDAMENTO = ['NOVO', 'AGUARDANDO', 'BLOQUEADO', 'EM ATENDIMENTO']

STATUS\_ATRASADO = ['ATRASADO']

STATUS\_ATIVO = ['ATIVO']

# Colunas obrigatórias

COLUNAS\_OBRIGATORIAS = ['Projeto', 'Status', 'Squad', 'Faturamento', 'HorasTrabalhadas']

COLUNAS\_NUMERICAS = ['Horas', 'HorasRestantes', 'Conclusao', 'HorasTrabalhadas']

COLUNAS\_TEXTO = ['Squad', 'Status', 'Faturamento', 'Especialista', 'Account Manager']

# Configurações

HORAS\_POR\_PESSOA = 180 # horas/mês

PESSOAS\_POR\_SQUAD = 3 # pessoas por squad

CAPACIDADE\_TOTAL = HORAS\_POR\_PESSOA \* PESSOAS\_POR\_SQUAD

```

### 16.5 Módulo Macro (app/macro/)

#### 16.5.1 Routes (routes.py)

```python

from flask import Blueprint, render\_template, jsonify, request

from .services import MacroService

import logging

bp = Blueprint('macro', \_\_name\_\_)

service = MacroService()

logger = logging.getLogger(\_\_name\_\_)

@bp.route('/')

def dashboard():

"""Rota principal do dashboard macro"""

try:

dados = service.carregar\_dados()

if dados.empty:

return render\_template('macro/dashboard.html',

erro="Não foi possível carregar os dados",

metricas={})

metricas = service.obter\_metricas\_macro(dados)

return render\_template('macro/dashboard.html',

metricas=metricas)

except Exception as e:

logger.error(f"Erro ao carregar dashboard macro: {str(e)}")

return render\_template('macro/erro.html',

erro="Ocorreu um erro ao carregar o dashboard")

@bp.route('/api/especialistas')

def especialistas():

"""API para obter dados de especialistas"""

try:

dados = service.carregar\_dados()

if dados.empty:

return jsonify([])

especialistas = service.obter\_especialistas(dados)

return jsonify(especialistas)

except Exception as e:

logger.error(f"Erro ao obter especialistas: {str(e)}")

return jsonify([])

@bp.route('/api/accounts')

def accounts():

"""API para obter dados de accounts"""

try:

dados = service.carregar\_dados()

if dados.empty:

return jsonify([])

accounts = service.obter\_accounts(dados)

return jsonify(accounts)

except Exception as e:

logger.error(f"Erro ao obter accounts: {str(e)}")

return jsonify([])

```

#### 16.5.2 Services (services.py)

```python

from .base\_service import BaseService

import pandas as pd

import logging

from datetime import datetime, timedelta

import numpy as np

logger = logging.getLogger(\_\_name\_\_)

class MacroService(BaseService):

"""Serviço para o módulo macro"""

def \_\_init\_\_(self):

super().\_\_init\_\_()

def obter\_metricas\_macro(self, dados):

"""Obtém métricas para o dashboard macro"""

try:

if dados is None or dados.empty:

return {}

metricas = {

'total\_projetos': len(dados),

'projetos\_ativos': len(dados[dados['Status'] == 'ATIVO']),

'projetos\_criticos': len(dados[dados['Status'] == 'CRITICO']),

'projetos\_concluidos': len(dados[dados['Status'] == 'CONCLUIDO']),

'projetos\_em\_atendimento': len(dados[dados['Status'] == 'EM ATENDIMENTO'])

}

return metricas

except Exception as e:

logger.error(f"Erro ao obter métricas macro: {str(e)}")

return {}

def obter\_especialistas(self, dados):

"""Obtém dados de especialistas"""

try:

if dados is None or dados.empty:

return []

especialistas = dados['Especialista'].unique()

return [{'nome': esp} for esp in especialistas if esp]

except Exception as e:

logger.error(f"Erro ao obter especialistas: {str(e)}")

return []

def obter\_accounts(self, dados):

"""Obtém dados de accounts"""

try:

if dados is None or dados.empty:

return []

accounts = dados['Account Manager'].unique()

return [{'nome': acc} for acc in accounts if acc]

except Exception as e:

logger.error(f"Erro ao obter accounts: {str(e)}")

return []

```

#### 16.5.3 Constants (constants.py)

```python

# Status dos projetos

STATUS\_NAO\_ATIVOS = ['FECHADO', 'ENCERRADO', 'RESOLVIDO', 'CANCELADO']

STATUS\_EM\_ANDAMENTO = ['NOVO', 'AGUARDANDO', 'BLOQUEADO', 'EM ATENDIMENTO']

STATUS\_ATRASADO = ['ATRASADO']

STATUS\_ATIVO = ['ATIVO']

# Constantes para cálculos

DIAS\_POR\_MES = 30

HORAS\_POR\_DIA = 8

DIAS\_POR\_SEMANA = 5

# Limites para alertas

LIMITE\_HORAS\_NEGATIVAS = 0

LIMITE\_ATRASO\_DIAS = 7

LIMITE\_CONCLUSAO\_MINIMA = 30

# Cores para visualização

CORES\_STATUS = {

'NOVO': 'info',

'EM ATENDIMENTO': 'primary',

'AGUARDANDO': 'warning',

'ENCERRADO': 'success',

'RESOLVIDO': 'success',

'FECHADO': 'success',

'BLOQUEADO': 'dark',

'ATRASADO': 'warning',

'CANCELADO': 'danger'

}

# Cores para conclusão

CORES\_CONCLUSAO = {

'success': 90, # Verde

'info': 70, # Azul

'warning': 50, # Amarelo

'danger': 0 # Vermelho

}

# Colunas obrigatórias

COLUNAS\_OBRIGATORIAS = ['Projeto', 'Status', 'Squad', 'Faturamento', 'HorasTrabalhadas']

COLUNAS\_NUMERICAS = ['Horas', 'HorasRestantes', 'Conclusao', 'HorasTrabalhadas']

COLUNAS\_TEXTO = ['Squad', 'Status', 'Faturamento', 'Especialista', 'Account Manager']

```

### 16.6 Módulo Micro (app/micro/)

#### 16.6.1 Routes (routes.py)

```python

from flask import Blueprint, render\_template, jsonify, request

from .services import MicroService

import logging

bp = Blueprint('micro', \_\_name\_\_)

service = MicroService()

logger = logging.getLogger(\_\_name\_\_)

@bp.route('/')

def dashboard():

"""Rota principal do dashboard micro"""

try:

dados = service.carregar\_dados()

if dados.empty:

return render\_template('micro/dashboard.html',

erro="Não foi possível carregar os dados",

metricas={})

metricas = service.obter\_metricas\_micro(dados)

return render\_template('micro/dashboard.html',

metricas=metricas)

except Exception as e:

logger.error(f"Erro ao carregar dashboard micro: {str(e)}")

return render\_template('micro/erro.html',

erro="Ocorreu um erro ao carregar o dashboard")

@bp.route('/api/projetos/especialista/<path:nome\_especialista>')

def projetos\_por\_especialista(nome\_especialista):

"""API para obter projetos por especialista"""

try:

dados = service.carregar\_dados()

if dados.empty:

return jsonify([])

projetos = service.obter\_projetos\_por\_especialista(dados, nome\_especialista)

return jsonify(projetos)

except Exception as e:

logger.error(f"Erro ao obter projetos por especialista: {str(e)}")

return jsonify([])

@bp.route('/api/projetos/account/<path:nome\_account>')

def projetos\_por\_account(nome\_account):

"""API para obter projetos por account"""

try:

dados = service.carregar\_dados()

if dados.empty:

return jsonify([])

projetos = service.obter\_projetos\_por\_account(dados, nome\_account)

return jsonify(projetos)

except Exception as e:

logger.error(f"Erro ao obter projetos por account: {str(e)}")

return jsonify([])

```

#### 16.6.2 Services (services.py)

```python

from app.utils.base\_service import BaseService

from app.utils.constants import (

STATUS\_ATIVO,

STATUS\_CRITICO,

STATUS\_CONCLUIDO,

STATUS\_ATENDIMENTO

)

import pandas as pd

import logging

from datetime import datetime, timedelta

logger = logging.getLogger(\_\_name\_\_)

class MicroService(BaseService):

"""Serviço para o módulo micro"""

def \_\_init\_\_(self):

super().\_\_init\_\_()

def obter\_metricas\_micro(self, dados):

"""Obtém métricas para o dashboard micro"""

try:

if dados is None or dados.empty:

return {}

metricas = {

'total\_projetos': len(dados),

'projetos\_ativos': len(dados[dados['Status'] == STATUS\_ATIVO]),

'projetos\_criticos': len(dados[dados['Status'] == STATUS\_CRITICO]),

'projetos\_concluidos': len(dados[dados['Status'] == STATUS\_CONCLUIDO]),

'projetos\_em\_atendimento': len(dados[dados['Status'] == STATUS\_ATENDIMENTO])

}

return metricas

except Exception as e:

logger.error(f"Erro ao obter métricas micro: {str(e)}")

return {}

def obter\_projetos\_por\_especialista(self, dados, nome\_especialista):

"""Obtém projetos por especialista"""

try:

if dados is None or dados.empty:

return []

projetos = dados[dados['Especialista'] == nome\_especialista]

return self.\_formatar\_projetos(projetos)

except Exception as e:

logger.error(f"Erro ao obter projetos por especialista: {str(e)}")

return []

def obter\_projetos\_por\_account(self, dados, nome\_account):

"""Obtém projetos por account manager"""

try:

if dados is None or dados.empty:

return []

projetos = dados[dados['Account Manager'] == nome\_account]

return self.\_formatar\_projetos(projetos)

except Exception as e:

logger.error(f"Erro ao obter projetos por account: {str(e)}")

return []

def obter\_projetos\_ativos(self, dados):

"""Obtém projetos ativos"""

try:

if dados is None or dados.empty:

return []

projetos = dados[dados['Status'] == STATUS\_ATIVO]

return self.\_formatar\_projetos(projetos)

except Exception as e:

logger.error(f"Erro ao obter projetos ativos: {str(e)}")

return []

def obter\_projetos\_criticos(self, dados):

"""Obtém projetos críticos"""

try:

if dados is None or dados.empty:

return []

projetos = dados[dados['Status'] == STATUS\_CRITICO]

return self.\_formatar\_projetos(projetos)

except Exception as e:

logger.error(f"Erro ao obter projetos críticos: {str(e)}")

return []

def obter\_projetos\_concluidos(self, dados):

"""Obtém projetos concluídos"""

try:

if dados is None or dados.empty:

return []

projetos = dados[dados['Status'] == STATUS\_CONCLUIDO]

return self.\_formatar\_projetos(projetos)

except Exception as e:

logger.error(f"Erro ao obter projetos concluídos: {str(e)}")

return []

```

#### 16.6.3 Constants (constants.py)

```python

# Status dos projetos

STATUS\_NAO\_ATIVOS = ['FECHADO', 'ENCERRADO', 'RESOLVIDO', 'CANCELADO']

STATUS\_EM\_ANDAMENTO = ['NOVO', 'AGUARDANDO', 'BLOQUEADO', 'EM ATENDIMENTO']

STATUS\_ATRASADO = ['ATRASADO']

STATUS\_ATIVO = ['ATIVO']

STATUS\_CRITICO = ['CRITICO']

STATUS\_CONCLUIDO = ['CONCLUIDO']

# Constantes para cálculos

DIAS\_POR\_MES = 30

HORAS\_POR\_DIA = 8

DIAS\_POR\_SEMANA = 5

# Limites para alertas

LIMITE\_HORAS\_NEGATIVAS = 0

LIMITE\_ATRASO\_DIAS = 7

LIMITE\_CONCLUSAO\_MINIMA = 30

# Cores para visualização

CORES\_STATUS = {

'NOVO': 'info',

'EM ATENDIMENTO': 'primary',

'AGUARDANDO': 'warning',

'ENCERRADO': 'success',

'RESOLVIDO': 'success',

'FECHADO': 'success',

'BLOQUEADO': 'dark',

'ATRASADO': 'warning',

'CANCELADO': 'danger',

'CRITICO': 'danger',

'CONCLUIDO': 'success'

}

# Cores para conclusão

CORES\_CONCLUSAO = {

'success': 90, # Verde

'info': 70, # Azul

'warning': 50, # Amarelo

'danger': 0 # Vermelho

}

# Colunas obrigatórias

COLUNAS\_OBRIGATORIAS = ['Projeto', 'Status', 'Squad', 'Faturamento', 'HorasTrabalhadas']

COLUNAS\_NUMERICAS = ['Horas', 'HorasRestantes', 'Conclusao', 'HorasTrabalhadas']

COLUNAS\_TEXTO = ['Squad', 'Status', 'Faturamento', 'Especialista', 'Account Manager']

```

## 17. Arquivos Estáticos, Templates e App Principal

### 17.1 Arquivos Estáticos (static/)

```

static/

├── css/

│ ├── main.css # Estilos principais

│ ├── dashboard.css # Estilos dos dashboards

│ └── components.css # Estilos dos componentes

├── js/

│ ├── main.js # Scripts principais

│ ├── charts.js # Configuração de gráficos

│ └── utils.js # Funções utilitárias

└── img/

├── logo.png # Logo da aplicação

└── icons/ # Ícones do sistema

```

#### 17.1.1 CSS Principal (static/css/main.css)

```css

:root {

--primary-color: #007bff;

--secondary-color: #6c757d;

--success-color: #28a745;

--danger-color: #dc3545;

--warning-color: #ffc107;

--info-color: #17a2b8;

--light-color: #f8f9fa;

--dark-color: #343a40;

}

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

line-height: 1.6;

color: #333;

background-color: #f5f5f5;

}

.container {

max-width: 1200px;

margin: 0 auto;

padding: 20px;

}

.card {

background: #fff;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

padding: 20px;

margin-bottom: 20px;

}

.btn {

padding: 8px 16px;

border-radius: 4px;

border: none;

cursor: pointer;

transition: all 0.3s ease;

}

.btn-primary {

background-color: var(--primary-color);

color: white;

}

.table {

width: 100%;

border-collapse: collapse;

margin-bottom: 1rem;

}

.table th,

.table td {

padding: 12px;

border-bottom: 1px solid #dee2e6;

}

.status-badge {

padding: 4px 8px;

border-radius: 12px;

font-size: 0.875rem;

font-weight: 500;

}

.status-active { background-color: var(--success-color); color: white; }

.status-warning { background-color: var(--warning-color); color: black; }

.status-danger { background-color: var(--danger-color); color: white; }

```

#### 17.1.2 JavaScript Principal (static/js/main.js)

```javascript

// Configuração de gráficos

const chartConfig = {

type: 'line',

options: {

responsive: true,

maintainAspectRatio: false,

scales: {

y: {

beginAtZero: true

}

}

}

};

// Funções de utilidade

const utils = {

formatDate: (date) => {

return new Date(date).toLocaleDateString('pt-BR');

},

formatNumber: (number) => {

return new Intl.NumberFormat('pt-BR').format(number);

},

formatCurrency: (value) => {

return new Intl.NumberFormat('pt-BR', {

style: 'currency',

currency: 'BRL'

}).format(value);

}

};

// Manipulação de eventos

document.addEventListener('DOMContentLoaded', () => {

// Inicialização de tooltips

const tooltips = document.querySelectorAll('[data-toggle="tooltip"]');

tooltips.forEach(tooltip => {

new bootstrap.Tooltip(tooltip);

});

// Inicialização de gráficos

const charts = document.querySelectorAll('.chart');

charts.forEach(chart => {

const ctx = chart.getContext('2d');

new Chart(ctx, {

...chartConfig,

data: JSON.parse(chart.dataset.chartData)

});

});

});

```

### 17.2 Templates (templates/)

```

templates/

├── base.html # Template base

├── gerencial/

│ ├── dashboard.html # Dashboard gerencial

│ ├── erro.html # Página de erro

│ └── partials/

│ ├── filtros.html # Componente de filtros

│ └── tabela.html # Componente de tabela

├── macro/

│ ├── dashboard.html # Dashboard macro

│ ├── erro.html # Página de erro

│ └── partials/

│ ├── graficos.html # Componente de gráficos

│ └── metricas.html # Componente de métricas

└── micro/

├── dashboard.html # Dashboard micro

├── erro.html # Página de erro

└── partials/

├── detalhes.html # Componente de detalhes

└── status.html # Componente de status

```

#### 17.2.1 Template Base (templates/base.html)

```html

<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>{% block title %}Control360 SOU{% endblock %}</title>

<!-- CSS -->

<link rel="stylesheet" href="{{ url\_for('static', filename='css/main.css') }}">

<link rel="stylesheet" href="{{ url\_for('static', filename='css/dashboard.css') }}">

<link rel="stylesheet" href="{{ url\_for('static', filename='css/components.css') }}">

<!-- JavaScript -->

<script src="{{ url\_for('static', filename='js/main.js') }}" defer></script>

<script src="{{ url\_for('static', filename='js/charts.js') }}" defer></script>

<script src="{{ url\_for('static', filename='js/utils.js') }}" defer></script>

</head>

<body>

<!-- Header -->

<header class="main-header">

<div class="container">

<nav class="main-nav">

<a href="{{ url\_for('gerencial.dashboard') }}" class="logo">

<img src="{{ url\_for('static', filename='img/logo.png') }}" alt="Control360 SOU">

</a>

<ul class="nav-links">

<li><a href="{{ url\_for('gerencial.dashboard') }}">Gerencial</a></li>

<li><a href="{{ url\_for('macro.dashboard') }}">Macro</a></li>

<li><a href="{{ url\_for('micro.dashboard') }}">Micro</a></li>

</ul>

</nav>

</div>

</header>

<!-- Main Content -->

<main class="main-content">

<div class="container">

{% block content %}{% endblock %}

</div>

</main>

<!-- Footer -->

<footer class="main-footer">

<div class="container">

<p>&copy; 2024 Control360 SOU. Todos os direitos reservados.</p>

</div>

</footer>

</body>

</html>

```

#### 17.2.2 Dashboard Gerencial (templates/gerencial/dashboard.html)

```html

{% extends "base.html" %}

{% block title %}Dashboard Gerencial - Control360 SOU{% endblock %}

{% block content %}

<div class="dashboard-container">

<!-- Filtros -->

{% include 'gerencial/partials/filtros.html' %}

<!-- Métricas Principais -->

<div class="metrics-grid">

<div class="card metric-card">

<h3>Total de Projetos</h3>

<p class="metric-value">{{ metricas.total\_projetos }}</p>

</div>

<div class="card metric-card">

<h3>Projetos Ativos</h3>

<p class="metric-value">{{ metricas.projetos\_ativos }}</p>

</div>

<div class="card metric-card">

<h3>Projetos Críticos</h3>

<p class="metric-value">{{ metricas.projetos\_criticos }}</p>

</div>

<div class="card metric-card">

<h3>Taxa de Burn Rate</h3>

<p class="metric-value">{{ "%.1f"|format(metricas.burn\_rate) }}%</p>

</div>

</div>

<!-- Gráficos -->

<div class="charts-grid">

<div class="card">

<h3>Projetos por Squad</h3>

<canvas class="chart" data-chart-data="{{ projetos\_por\_squad|tojson }}"></canvas>

</div>

<div class="card">

<h3>Projetos por Faturamento</h3>

<canvas class="chart" data-chart-data="{{ projetos\_por\_faturamento|tojson }}"></canvas>

</div>

</div>

<!-- Tabelas -->

<div class="tables-grid">

<div class="card">

<h3>Projetos Críticos</h3>

{% include 'gerencial/partials/tabela.html' with context %}

</div>

</div>

</div>

{% endblock %}

```

#### 17.2.3 Dashboard Macro (templates/macro/dashboard.html)

```html

{% extends "base.html" %}

{% block title %}Dashboard Macro - Control360 SOU{% endblock %}

{% block content %}

<div class="dashboard-container">

<!-- Filtros -->

{% include 'macro/partials/filtros.html' %}

<!-- Métricas Principais -->

<div class="metrics-grid">

<div class="card metric-card">

<h3>Projetos Ativos</h3>

<p class="metric-value">{{ metricas.projetos\_ativos }}</p>

</div>

<div class="card metric-card">

<h3>Projetos Críticos</h3>

<p class="metric-value">{{ metricas.projetos\_criticos }}</p>

</div>

<div class="card metric-card">

<h3>Média de Horas</h3>

<p class="metric-value">{{ "%.1f"|format(metricas.media\_horas) }}</p>

</div>

<div class="card metric-card">

<h3>Eficiência de Entrega</h3>

<p class="metric-value">{{ "%.1f"|format(metricas.eficiencia\_entrega) }}%</p>

</div>

</div>

<!-- Gráficos -->

<div class="charts-grid">

<div class="card">

<h3>Projetos por Status</h3>

<canvas class="chart" data-chart-data="{{ projetos\_por\_status|tojson }}"></canvas>

</div>

<div class="card">

<h3>Projetos por Especialista</h3>

<canvas class="chart" data-chart-data="{{ projetos\_por\_especialista|tojson }}"></canvas>

</div>

</div>

<!-- Tabelas -->

<div class="tables-grid">

<div class="card">

<h3>Especialistas</h3>

{% include 'macro/partials/tabela\_especialistas.html' with context %}

</div>

<div class="card">

<h3>Accounts</h3>

{% include 'macro/partials/tabela\_accounts.html' with context %}

</div>

</div>

</div>

{% endblock %}

```

#### 17.2.4 Partials do Módulo Macro

##### 17.2.4.1 Filtros (templates/macro/partials/filtros.html)

```html

<div class="filters-container">

<form class="filters-form" method="GET">

<div class="form-group">

<label for="squad">Squad:</label>

<select name="squad" id="squad" class="form-control">

<option value="">Todos</option>

{% for squad in squads %}

<option value="{{ squad }}" {% if squad == filtro\_aplicado.squad %}selected{% endif %}>

{{ squad }}

</option>

{% endfor %}

</select>

</div>

<div class="form-group">

<label for="especialista">Especialista:</label>

<select name="especialista" id="especialista" class="form-control">

<option value="">Todos</option>

{% for esp in especialistas %}

<option value="{{ esp }}" {% if esp == filtro\_aplicado.especialista %}selected{% endif %}>

{{ esp }}

</option>

{% endfor %}

</select>

</div>

<div class="form-group">

<label for="data">Data:</label>

<input type="date" name="data" id="data" class="form-control"

value="{{ filtro\_aplicado.data }}">

</div>

<button type="submit" class="btn btn-primary">Aplicar Filtros</button>

</form>

</div>

```

##### 17.2.4.2 Tabela de Especialistas (templates/macro/partials/tabela\_especialistas.html)

```html

<div class="table-responsive">

<table class="table">

<thead>

<tr>

<th>Especialista</th>

<th>Squad</th>

<th>Projetos Ativos</th>

<th>Taxa de Uso</th>

<th>Horas Disponíveis</th>

</tr>

</thead>

<tbody>

{% for esp in especialistas %}

<tr>

<td>{{ esp.nome }}</td>

<td>{{ esp.squad }}</td>

<td>{{ esp.projetos\_ativos }}</td>

<td>

<div class="progress">

<div class="progress-bar" role="progressbar"

style="width: {{ esp.taxa\_uso }}%"

aria-valuenow="{{ esp.taxa\_uso }}"

aria-valuemin="0"

aria-valuemax="100">

{{ "%.1f"|format(esp.taxa\_uso) }}%

</div>

</div>

</td>

<td>{{ "%.1f"|format(esp.horas\_disponiveis) }}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

```

##### 17.2.4.3 Tabela de Accounts (templates/macro/partials/tabela\_accounts.html)

```html

<div class="table-responsive">

<table class="table">

<thead>

<tr>

<th>Account Manager</th>

<th>Projetos</th>

<th>Faturamento Total</th>

<th>Projetos Ativos</th>

<th>Satisfação</th>

</tr>

</thead>

<tbody>

{% for acc in accounts %}

<tr>

<td>{{ acc.nome }}</td>

<td>{{ acc.total\_projetos }}</td>

<td>{{ "R$ %.2f"|format(acc.faturamento\_total) }}</td>

<td>{{ acc.projetos\_ativos }}</td>

<td>

<div class="progress">

<div class="progress-bar" role="progressbar"

style="width: {{ acc.satisfacao }}%"

aria-valuenow="{{ acc.satisfacao }}"

aria-valuemin="0"

aria-valuemax="100">

{{ "%.1f"|format(acc.satisfacao) }}%

</div>

</div>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

```

#### 17.2.5 Partials do Módulo Gerencial

##### 17.2.5.1 Filtros (templates/gerencial/partials/filtros.html)

```html

<div class="filters-container">

<form class="filters-form" method="GET">

<div class="form-group">

<label for="squad">Squad:</label>

<select name="squad" id="squad" class="form-control">

<option value="">Todos</option>

{% for squad in squads %}

<option value="{{ squad }}" {% if squad == filtro\_aplicado.squad %}selected{% endif %}>

{{ squad }}

</option>

{% endfor %}

</select>

</div>

<div class="form-group">

<label for="faturamento">Faturamento:</label>

<select name="faturamento" id="faturamento" class="form-control">

<option value="">Todos</option>

{% for fat in faturamentos %}

<option value="{{ fat }}" {% if fat == filtro\_aplicado.faturamento %}selected{% endif %}>

{{ fat }}

</option>

{% endfor %}

</select>

</div>

<button type="submit" class="btn btn-primary">Aplicar Filtros</button>

</form>

</div>

```

##### 17.2.5.2 Tabela de Projetos (templates/gerencial/partials/tabela.html)

```html

<div class="table-responsive">

<table class="table">

<thead>

<tr>

<th>Projeto</th>

<th>Squad</th>

<th>Status</th>

<th>Conclusão</th>

<th>Horas Restantes</th>

<th>Vencimento</th>

</tr>

</thead>

<tbody>

{% for projeto in projetos %}

<tr>

<td>{{ projeto.Projeto }}</td>

<td>{{ projeto.Squad }}</td>

<td>

<span class="status-badge status-{{ projeto.Status|lower }}">

{{ projeto.Status }}

</span>

</td>

<td>

<div class="progress">

<div class="progress-bar" role="progressbar"

style="width: {{ projeto.Conclusao }}%"

aria-valuenow="{{ projeto.Conclusao }}"

aria-valuemin="0"

aria-valuemax="100">

{{ "%.1f"|format(projeto.Conclusao) }}%

</div>

</div>

</td>

<td>{{ "%.1f"|format(projeto.HorasRestantes) }}</td>

<td>{{ projeto.VencimentoEm }}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

```

### 17.3 Arquivo Principal (app.py)

```python

from flask import Flask, redirect, url\_for

import logging

from logging.handlers import RotatingFileHandler

import os

from pathlib import Path

from app.utils.json\_provider import NumpyJSONProvider

# Configuração de diretórios

BASE\_DIR = Path(\_\_file\_\_).parent

LOG\_DIR = BASE\_DIR / 'logs'

LOG\_DIR.mkdir(parents=True, exist\_ok=True)

# Configuração de logging

def setup\_logging(app):

log\_file = LOG\_DIR / 'app.log'

log\_format = logging.Formatter(

'%(asctime)s - %(name)s - %(levelname)s - %(message)s [in %(pathname)s:%(lineno)d]'

)

file\_handler = RotatingFileHandler(

log\_file, maxBytes=1024\*1024\*5, backupCount=3, encoding='utf-8'

)

file\_handler.setFormatter(log\_format)

file\_handler.setLevel(logging.DEBUG)

stream\_handler = logging.StreamHandler()

stream\_handler.setFormatter(log\_format)

stream\_handler.setLevel(logging.DEBUG)

app.logger.addHandler(file\_handler)

app.logger.addHandler(stream\_handler)

app.logger.setLevel(logging.DEBUG)

# Criação da aplicação

def create\_app():

app = Flask(\_\_name\_\_,

template\_folder=str(BASE\_DIR / 'templates'),

static\_folder=str(BASE\_DIR / 'static'))

# Configuração do JSON provider

app.json = NumpyJSONProvider(app)

# Setup de logging

setup\_logging(app)

# Registro de blueprints

from app.gerencial import gerencial\_bp

from app.macro import macro\_bp

from app.micro import micro\_bp

app.register\_blueprint(gerencial\_bp, url\_prefix='/gerencial')

app.register\_blueprint(macro\_bp, url\_prefix='/macro')

app.register\_blueprint(micro\_bp, url\_prefix='/micro')

# Rota principal

@app.route('/')

def index():

return redirect(url\_for('gerencial.dashboard'))

# Verificação de templates

check\_templates(app)

return app

# Verificação de templates essenciais

def check\_templates(app):

required\_templates = [

'gerencial/dashboard.html',

'gerencial/erro.html',

'macro/dashboard.html',

'micro/dashboard.html',

]

for template in required\_templates:

if not template\_exists(app, template):

app.logger.warning(f"Template essencial não encontrado: {template}")

# Inicialização da aplicação

if \_\_name\_\_ == '\_\_main\_\_':

app = create\_app()

app.run(debug=True, host='0.0.0.0', port=5000)

```