

DOCUMENTAÇÃO TÉCNICA

Sistema Concurso Vision

Tecnologias: Python 3.9+, Flask 3.0, MySQL 8.0, PyMySQL

Autores: Janary Victor do Nascimento Júnior - RA: 1362416604

José Kleivson da Silva - RA: 1362411072

Carlos Henrique Cavalcanti Moreira – RA 1362416272

PARTE 1 - MANUAL DO DESENVOLVEDOR

1. VISÃO GERAL DA ARQUITETURA

1.1 Arquitetura do Sistema

O Concurso Vision utiliza arquitetura MVC (Model-View-Controller) implementada com Flask:

- **Model:** Banco de dados MySQL com tabelas usuarios e concursos
- **View:** Templates HTML renderizados com Jinja2
- **Controller:** Rotas Flask definidas em app.py

1.2 Stack Tecnológico

Camada	Tecnologia	Versão
Backend	Python	3.9+
Framework Web	Flask	3.0.0
Banco de Dados	MySQL	8.0+
Conector DB	PyMySQL	1.1.0
Template Engine	Jinja2	3.1.2
Segurança	Werkzeug	3.0.1
Frontend	HTML5/CSS3/JS	-
Criptografia	scrypt (Werkzeug)	-

2. ESTRUTURA DO PROJETO

concurso-vision/

```
|
|
|— app.py          # Aplicação principal Flask
|— requirements.txt # Dependências Python
|— database.sql    # Script criação do BD
|— README.md      # Documentação básica
|— .gitignore     # Arquivos ignorados pelo Git
|
```

```
├── templates/          # Templates HTML
|   ├── base.html       # Template base
|   ├── login.html      # Tela de login
|   ├── cadastro.html   # Tela de cadastro
|   ├── home.html       # Página inicial
|   ├── concurso_detalhes.html # Detalhes do concurso
|   ├── admin_adicionar.html # Adicionar concurso
|   └── admin_editar.html # Editar concurso
|
├── static/             # Arquivos estáticos
|   ├── css/
|   |   └── style.css    # Estilos CSS
|   └── js/
|       └── script.js    # Scripts JavaScript
|
├── venv/               # Ambiente virtual (não versionar)
|
└── tests/              # Testes automatizados
    ├── test_auth.py
    ├── test_concursos.py
    └── test_admin.py
```

3. CONFIGURAÇÃO DO AMBIENTE

3.1 Requisitos do Sistema

- **Sistema Operacional:** Windows 10+, Linux, macOS
- **Python:** 3.9 ou superior
- **MySQL:** 5.7 ou superior
- **Memória RAM:** 2GB mínimo
- **Espaço em Disco:** 500MB

3.2 Instalação Passo a Passo

Passo 1: Clonar Repositório

git clone <https://github.com/seu-usuario/concurso-vision.git>

```
cd concurso-vision
```

Passo 2: Criar Ambiente Virtual

```
python -m venv venv
```

Passo 3: Ativar Ambiente Virtual

Windows:

```
venv\Scripts\activate
```

Linux/macOS:

```
source venv/bin/activate
```

Passo 4: Instalar Dependências

```
pip install -r requirements.txt
```

Passo 5: Configurar Banco de Dados

```
# Acessar MySQL
```

```
mysql -u root -p
```

```
# Executar script
```

```
source database.sql
```

Passo 6: Configurar Credenciais

Editar app.py linhas 11-16:

```
DB_CONFIG = {  
    'host': 'localhost',  
    'user': 'seu_usuario',  
    'password': 'sua_senha',  
    'database': 'concurso_vision',  
    'cursorclass': pymysql.cursors.DictCursor  
}
```

Passo 7: Executar Aplicação

```
python app.py
```

Acesse: <http://127.0.0.1:5000>

4. ARQUITETURA DE CÓDIGO

4.1 Arquivo Principal (app.py)

4.1.1 Importações

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
import pymysql

from werkzeug.security import generate_password_hash, check_password_hash

from datetime import datetime
```

4.1.2 Configuração do Flask

```
app = Flask(__name__)

app.secret_key = 'chave_secreta_forte' # Mudar em produção
```

4.1.3 Configuração do Banco de Dados

```
DB_CONFIG = {

    'host': 'localhost',

    'user': 'root',

    'password': 'senha',

    'database': 'concurso_vision',

    'cursorclass': pymysql.cursors.DictCursor

}
```

```
def get_db_connection():

    return pymysql.connect(**DB_CONFIG)
```

4.2 Estrutura de Rotas

Rota	Método	Descrição	Autenticação
/	GET	Redireciona para login ou home	-
/login	GET, POST	Página de login	Não
/cadastro	GET, POST	Página de cadastro	Não
/home	GET	Página inicial com concursos	Sim
/concurso/<id>	GET	Detalhes de um concurso	Sim
/admin/adicionar	GET, POST	Adicionar concurso	Admin
/admin/editar/<id>	GET, POST	Editar concurso	Admin
/admin/deletar/<id>	GET	Deletar concurso	Admin
/logout	GET	Encerrar sessão	Sim

4.3 Exemplo de Rota Completa

```
@app.route('/login', methods=['GET', 'POST'])

def login():

    # Verifica se é POST (envio de formulário)

    if request.method == 'POST':
```

```
# Obtém dados do formulário

email = request.form['email']

password = request.form['password']


# Conecta ao banco

conn = get_db_connection()

cursor = conn.cursor()


# Busca usuário

cursor.execute('SELECT * FROM usuarios WHERE email = %s', (email,))

user = cursor.fetchone()


# Fecha conexão

cursor.close()

conn.close()


# Valida credenciais

if user and check_password_hash(user['senha'], password):

    # Cria sessão

    session['loggedin'] = True

    session['id'] = user['id']

    session['nome'] = user['nome']

    session['email'] = user['email']

    session['is_admin'] = user['is_admin']


    # Redireciona

    return redirect(url_for('home'))

else:

    # Mensagem de erro

    flash('Email ou senha incorretos!', 'error')
```

```
# Renderiza template  
return render_template('login.html')
```

5. BANCO DE DADOS

5.1 Esquema do Banco

Tabela: usuarios

```
CREATE TABLE usuarios (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(100) NOT NULL,  
    email VARCHAR(100) NOT NULL UNIQUE,  
    senha VARCHAR(255) NOT NULL,  
    is_admin BOOLEAN DEFAULT FALSE,  
    data_criacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

Campos:

- id: Identificador único
- nome: Nome completo do usuário
- email: Email único para login
- senha: Hash da senha (scrypt)
- is_admin: Flag de permissão administrativa
- data_criacao: Data de criação do registro

Tabela: concursos

```
CREATE TABLE concursos (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(200) NOT NULL,  
    orgao VARCHAR(150) NOT NULL,  
    vagas INT,  
    escolaridade VARCHAR(100),  
    status ENUM('disponivel', 'previsto', 'terminado') DEFAULT 'previsto',  
    link_edital TEXT,  
    data_inscricao_inicio DATE,
```

```
data_inscricao_fim DATE,  
data_prova DATE,  
descricao TEXT,  
data_criacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
data_atualizacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP  
);
```

Campos:

- id: Identificador único
- nome: Nome do concurso
- orgao: Órgão responsável
- vagas: Número de vagas
- escolaridade: Nível mínimo exigido
- status: Estado do concurso (ENUM)
- link_edital: URL do edital oficial
- data_inscricao_inicio: Início das inscrições
- data_inscricao_fim: Fim das inscrições
- data_prova: Data da prova
- descricao: Descrição detalhada
- data_criacao: Data de criação
- data_atualizacao: Última modificação

5.2 Consultas SQL Principais

Buscar Concursos por Status

```
SELECT * FROM concursos  
WHERE status = 'disponivel'  
ORDER BY data_inscricao_inicio DESC;
```

Buscar Usuário por Email

```
SELECT * FROM usuarios  
WHERE email = ?;
```

Inserir Novo Concurso

```
INSERT INTO concursos
```

```
(nome, orgao, vagas, escolaridade, status, link_edital,  
data_inscricao_inicio, data_inscricao_fim, data_prova, descricao)  
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
```

Atualizar Concurso

```
UPDATE concursos  
  
SET nome=?, orgao=?, vagas=?, escolaridade=?, status=?,  
    link_edital=?, data_inscricao_inicio=?, data_inscricao_fim=?,  
    data_prova=?, descricao=?  
  
WHERE id=?;
```

Deletar Concurso

```
DELETE FROM concursos WHERE id = ?;
```

6. SEGURANÇA

6.1 Autenticação

Hash de Senha:

- Utiliza Werkzeug Security com algoritmo scrypt
- Salt aleatório gerado automaticamente
- Hash armazenado com formato: scrypt:32768:8:1\$salt\$hash

```
# Gerar hash
```

```
hashed = generate_password_hash('senha123')
```

```
# Verificar senha
```

```
is_valid = check_password_hash(hashed, 'senha123')
```

6.2 Sessões

Gerenciamento:

- Secret key configurável em app.secret_key
- Dados armazenados em cookies criptografados
- Sessão expira ao fechar navegador

```
# Criar sessão
```

```
session['logged_in'] = True
```

```
session['id'] = user['id']
```



```
# Verificar sessão
```

```
if 'loggedin' in session:
```

```
    # Usuário autenticado
```

6.3 Proteção de Rotas

Controle de Acesso:

```
# Verificar autenticação
```

```
if 'loggedin' not in session:
```

```
    return redirect(url_for('login'))
```

```
# Verificar admin
```

```
if not session.get('is_admin'):
```

```
    flash('Acesso negado!', 'error')
```

```
    return redirect(url_for('home'))
```

6.4 Prevenção de SQL Injection

Queries Parametrizadas:

```
# CORRETO - Usa placeholders
```

```
cursor.execute('SELECT * FROM usuarios WHERE email = %s', (email,))
```

```
# ERRADO - Vulnerável a SQL Injection
```

```
cursor.execute(f'SELECT * FROM usuarios WHERE email = "{email}"')
```

6.5 Validação de Entrada

Validação no Backend:

```
# Validar campos obrigatórios
```

```
if not nome or not orgao or not status:
```

```
    flash('Campos obrigatórios não preenchidos!', 'error')
```

```
    return redirect(url_for('adicionar_concurso'))
```

7. FRONTEND

7.1 Templates Jinja2

Estrutura Base (base.html):

```
<!DOCTYPE html>

<html lang="pt-BR">

<head>

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

  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>

  {% block content %}{% endblock %}

  <script src="{{ url_for('static', filename='js/script.js') }}"></script>

</body>

</html>
```

Herança de Templates:

```
{% extends "base.html" %}

{% block title %}Login - Concurso Vision{% endblock %}
```

```
{% block content %}

  <!-- Conteúdo específico -->

{% endblock %}
```

Variáveis Jinja2:

```
<!-- Exibir variável -->

{{ concurso.nome }}

<!-- Condicional -->

{% if is_admin %}

  <a href="/admin/adicionar">Adicionar</a>

{% endif %}
```

```
<!-- Loop -->

{% for concurso in disponiveis %}

  <h3>{{ concurso.nome }}</h3>
```

```
{% endfor %}
```

7.2 CSS (style.css)

Variáveis CSS:

```
:root {  
  --primary: #2563eb;  
  --success: #10b981;  
  --warning: #f59e0b;  
  --danger: #ef4444;  
}
```

Responsividade:

```
@media (max-width: 768px) {  
  .concursos-grid {  
    grid-template-columns: 1fr;  
  }  
}
```

7.3 JavaScript (script.js)

Validação de Formulários:

```
forms.forEach(form => {  
  form.addEventListener('submit', function(e) {  
    const requiredFields = form.querySelectorAll('[required]');  
    let isValid = true;  
  
    requiredFields.forEach(field => {  
      if (!field.value.trim()) {  
        isValid = false;  
        field.style.borderColor = '#ef4444';  
      }  
    });  
  
    if (!isValid) {  
      e.preventDefault();  
    }  
  });  
});
```

```
        alert('Preencha todos os campos obrigatórios!');
    }
});
});
```

8. TESTES

8.1 Estrutura de Testes

```
# tests/test_auth.py
```

```
import unittest
```

```
from app import app
```

```
class TestAuth(unittest.TestCase):
```

```
    def setUp(self):
```

```
        self.app = app.test_client()
```

```
        self.app.testing = True
```

```
    def test_login_page(self):
```

```
        response = self.app.get('/login')
```

```
        self.assertEqual(response.status_code, 200)
```

```
    def test_login_success(self):
```

```
        response = self.app.post('/login', data={
```

```
            'email': 'admin@concursovision.com',
```

```
            'password': 'admin123'
```

```
        }, follow_redirects=True)
```

```
        self.assertEqual(response.status_code, 200)
```

8.2 Executar Testes

```
# Instalar pytest
```

```
pip install pytest
```

```
# Executar todos os testes
```

```
pytest tests/
```

```
# Executar teste específico
```

```
pytest tests/test_auth.py
```

```
# Com cobertura
```

```
pip install pytest-cov
```

```
pytest --cov=app tests/
```

9. DEPLOY

9.1 Preparação para Produção

Checklist:

- [] Alterar app.secret_key para valor forte e único
- [] Configurar debug=False
- [] Usar variáveis de ambiente para credenciais
- [] Configurar HTTPS
- [] Implementar rate limiting
- [] Configurar logs
- [] Backup automático do banco

Exemplo com Variáveis de Ambiente:

```
import os
```

```
app.secret_key = os.environ.get('SECRET_KEY', 'dev-key')
```

```
app.config['DEBUG'] = os.environ.get('DEBUG', 'False') == 'True'
```

```
DB_CONFIG = {
```

```
    'host': os.environ.get('DB_HOST', 'localhost'),
```

```
    'user': os.environ.get('DB_USER', 'root'),
```

```
    'password': os.environ.get('DB_PASSWORD'),
```

```
    'database': os.environ.get('DB_NAME', 'concurso_vision')
```

```
}
```

9.2 Deploy com Gunicorn

Instalar Gunicorn

```
pip install gunicorn
```

Executar

```
gunicorn -w 4 -b 0.0.0.0:5000 app:app
```

9.3 Deploy com Docker

Dockerfile

```
FROM python:3.11-slim
```

```
WORKDIR /app
```

```
COPY requirements.txt .
```

```
RUN pip install --no-cache-dir -r requirements.txt
```

```
COPY . .
```

```
CMD ["gunicorn", "-w", "4", "-b", "0.0.0.0:5000", "app:app"]
```

10. MANUTENÇÃO

10.1 Logs

Configurar Logging:

```
import logging
```

```
logging.basicConfig(
```

```
    filename='app.log',
```

```
    level=logging.INFO,
```

```
    format='%(asctime)s - %(levelname)s - %(message)s'
```

```
)
```

Usar

```
logging.info('Usuário fez login: ' + email)
```

```
logging.error('Erro ao conectar banco: ' + str(e))
```

10.2 Backup do Banco

Criar backup

```
mysqldump -u root -p concurso_vision > backup.sql
```

Restaurar backup

```
mysql -u root -p concurso_vision < backup.sql
```

10.3 Atualização de Dependências

Listar pacotes desatualizados

```
pip list --outdated
```

Atualizar pacote específico

```
pip install --upgrade Flask
```

Atualizar requirements.txt

```
pip freeze > requirements.txt
```

11. SOLUÇÃO DE PROBLEMAS

11.1 Problemas Comuns

Erro: "Can't connect to MySQL server"

Solução:

1. Verificar se MySQL está rodando
2. Conferir credenciais em DB_CONFIG
3. Testar: `mysql -u root -p`

Erro: "Template not found"

Solução:

1. Verificar pasta templates/ existe
2. Conferir nome do arquivo (case-sensitive)
3. Verificar estrutura: templates/login.html

Erro: "Secret key not set"

Solução:

1. Definir app.secret_key no app.py
2. Usar valor forte e único

11.2 Debug

Ativar Modo Debug:

```
if __name__ == '__main__':  
    app.run(debug=True)
```

Imprimir Variáveis:

```
print(f"Usuário: {user}")  
print(f"Session: {session}")
```

PARTE 2 - GUIA DO USUÁRIO

1. INTRODUÇÃO

O Concurso Vision é uma plataforma web para consultar e gerenciar informações sobre concursos públicos. Este guia explica como usar todas as funcionalidades do sistema.

2. PRIMEIROS PASSOS

2.1 Acessar o Sistema

1. Abra seu navegador
2. Acesse: <http://localhost:5000> ou <https://concursovision.com>
3. Você verá a tela de login

2.2 Criar Conta

Passo a Passo:

1. Na tela de login, clique em "**Cadastre-se**"
2. Preencha os dados:
 - **Nome Completo:** Seu nome
 - **Email:** Email válido (será usado para login)
 - **Senha:** Mínimo 6 caracteres
3. Clique em "**Cadastrar**"
4. Você será redirecionado para o login

2.3 Fazer Login

Passo a Passo:

1. Digite seu **email**
2. Digite sua **senha**
3. Clique em "**Entrar**"
4. Você será direcionado para a página inicial

3. FUNCIONALIDADES PARA USUÁRIOS**3.1 Página Inicial**

Ao fazer login, você verá três seções:

Concursos Disponíveis

- Concursos com inscrições abertas
- Mostra: nome, órgão, vagas, escolaridade, período de inscrição
- Badge verde: "Disponível"

Concursos Previstos

- Concursos anunciados, sem edital publicado
- Informações podem estar incompletas
- Badge amarelo: "Previsto"

Concursos Encerrados

- Concursos finalizados
- Últimos 10 encerrados
- Badge cinza: "Encerrado"


3.2 Ver Detalhes de um Concurso**Passo a Passo:**

1. Na página inicial, localize o concurso
2. Clique em "**Ver Detalhes**"
3. Visualize todas as informações:
 - Órgão responsável
 - Número de vagas
 - Escolaridade mínima exigida
 - Status do concurso
 - Período de inscrições

- Data da prova
- Descrição completa
- Link para edital oficial

3.3 Acessar Edital

Passo a Passo:

1. Entre nos detalhes do concurso
2. Role até o final da página
3. Clique em " **Acessar Edital Completo**"
4. O edital abrirá em uma nova aba

Nota: Se o edital não estiver disponível, o botão não aparecerá.

3.4 Sair do Sistema

Passo a Passo:

1. Clique em "**Sair**" no canto superior direito
2. Você será desconectado e redirecionado para o login

4. FUNCIONALIDADES PARA ADMINISTRADORES

4.1 Identificar Perfil Admin

Usuários administradores veem:

- Botão verde "+ **Adicionar Concurso**" no topo
- Botões "**Editar**" em cada concurso
- Botões "**Deletar**" na página de detalhes

4.2 Adicionar Novo Concurso

Passo a Passo:

1. Clique em "+ **Adicionar Concurso**" (topo da página)
2. Preencha o formulário:

Campos Obrigatórios (*):

- Nome do Concurso
- Órgão
- Status (Previsto/Disponível/Encerrado)

Campos Opcionais:

- Número de Vagas

- Escolaridade Mínima
 - Link do Edital (URL completa)
 - Início das Inscrições
 - Fim das Inscrições
 - Data da Prova
 - Descrição
3. Clique em **"Adicionar Concurso"**
 4. Você verá uma mensagem de sucesso
 5. O concurso aparecerá na página inicial

Dicas:

- Para concursos previstos, deixe datas em branco
- URL do edital deve ser completa: <https://exemplo.com/edital.pdf>
- A descrição pode ser longa e detalhada

4.3 Editar Concurso

Passo a Passo:

1. Localize o concurso desejado
2. Clique em **"Editar"** (botão amarelo)
3. Modifique os campos desejados
4. Clique em **" Salvar Alterações"**
5. Você verá uma mensagem de sucesso
6. As alterações estarão visíveis imediatamente

Casos de Uso:

- Mudar status de "Previsto" para "Disponível"
- Adicionar link do edital quando for publicado
- Corrigir informações incorretas
- Atualizar número de vagas

4.4 Excluir Concurso

Passo a Passo:

1. Entre nos detalhes do concurso
2. Role até o final da página
3. Clique em **" Deletar Concurso"** (botão vermelho)

4. Confirme a exclusão no popup
5. O concurso será removido permanentemente

ATENÇÃO: Esta ação não pode ser desfeita!

5. DICAS E BOAS PRÁTICAS

5.1 Para Candidatos

Verificar regularmente: Acesse diariamente para ver novos concursos

Salvar editais: Baixe os editais para consulta offline

Atentar-se aos prazos: Marque as datas de inscrição

Consultar encerrados: Veja editais anteriores para estudar

5.2 Para Administradores

Manter atualizado: Atualize status dos concursos regularmente

Informações completas: Preencha todos os campos possíveis

Verificar links: Teste os links de editais antes de salvar

Descrições claras: Escreva descrições objetivas e informativas

Status correto: Use "Previsto" apenas para concursos sem edital

5.3 Organização por Status

Previsto:

- Use quando há apenas rumor ou anúncio oficial
- Não preencha datas se não forem confirmadas
- Adicione descrição sobre a fonte da informação

Disponível:

- Use quando edital for publicado
- Preencha todas as datas
- Adicione link do edital obrigatoriamente

Encerrado:

- Mude status quando inscrições terminarem
- Mantenha link do edital disponível
- Útil para consultas históricas

6. PERGUNTAS FREQUENTES (FAQ)

Q: Esqueci minha senha. Como recuperar?

R: Atualmente o sistema não tem recuperação de senha. Entre em contato com o administrador.

Q: Posso editar meu perfil?

R: Na versão atual, não é possível editar dados do perfil após cadastro.

Q: Como me tornar administrador?

R: Apenas o administrador principal pode conceder permissões. Solicite ao responsável pelo sistema.

Q: O sistema funciona no celular?

R: Sim! O site é responsivo e funciona em smartphones e tablets.

Q: Posso salvar concursos favoritos?

R: Esta funcionalidade não está disponível na versão atual (MVP).

Q: Como sei se um concurso foi atualizado?

R: O sistema mostra a data de atualização em cada concurso.

Q: Posso exportar lista de concursos?

R: Não na versão atual. Funcionalidade planejada para versão futura.

7. SUPORTE

7.1 Reportar Problemas

Se encontrar algum problema:

1. Anote a mensagem de erro exata
2. Tire uma captura de tela (se possível)
3. Descreva o que estava fazendo
4. Entre em contato com suporte

7.2 Sugestões

Sugestões de melhorias são bem-vindas! Entre em contato através de:

- Email: suporte@concursovision.com
- GitHub: Issues no repositório

8. GLOSSÁRIO

Concurso: Processo seletivo para cargo público

Edital: Documento oficial com regras do concurso

MVP: Minimum Viable Product (Produto Mínimo Viável)

Admin: Administrador do sistema

Session: Sessão de login do usuário

Status: Estado atual do concurso (previsto/disponível/terminado)