

Instrução

Orientativa para

Desenvolvimento de

Sistemas no IFPE

Instrução Orientativa para Desenvolvimento de Sistemas no IFPE

- [Introdução](#)
- [Documento de Visão](#)
- [Documento de Oficialização da Demanda](#)
- [Documento de Arquitetura](#)
- [Linguagens e Guias de Estilo](#)
- [Documentação](#)
- [Interface do Usuário](#)
- [Release Notes](#)
- [Relatórios](#)
- [Implantação \(Web\)](#)

Introdução

Este documento tem por finalidade descrever procedimentos, documentos necessários e boas práticas a serem adotadas em um projeto de software para que o produto seja adotado como solução **institucional**.

É recomendável que a equipe engajada em desenvolver um software institucional envolva a DADT desde o começo do processo a fim de garantir conformidade às instruções normativas e, por conseguinte, facilitar a implantação sistemática da solução.

Análise e Projeto

- DOD (opcional)
- Visão
- Arquitetura

Desenvolvimento

Define linguagens e tecnologias suportadas e seus respectivos guias de estilo a fim de garantir legibilidade e qualidade do código gerado. Além disso, também são apresentadas boas práticas para documentação do código com o objetivo de facilitar futuras manutenções. Por fim, é apresentada uma proposta de interface gráfica construção dos sistemas. Essa proposta deve ser ajustada em parceria com a ASCOM para cada sistema desenvolvido.

- Linguagens e Guias de Estilo
- Documentação
- Interface de Usuário
- Release Notes

Testes

Para assegurar a qualidade do software, torna-se necessário envolver o processo de testes desde o começo do desenvolvimento em diversos níveis: desde testes unitários até testes de aceitação. Recomenda-se o uso de ferramentas de integração contínua a fim de descobrir problemas o mais cedo possível e facilitar a geração de relatórios.

- Caixa-Branca
- Caixa-Preta
- Relatórios

Implantação

- Web

Documento de Visão

Checklist

- [] Validado pelo requerente (escopo e protótipo de telas)

Introdução

Finalidade do documento

A finalidade deste documento é especificar os requisitos relevantes dos usuários, assim como os limites e restrições evidentes que dão uma visão geral. Essa visão viabiliza a identificação e a produção de documentos e requisitos mais técnicos, assim como do próprio sistema. A visão serve como forma de permitir a compreensão, pelos participantes do projeto, do "o quê e por quê" o projeto existe e provê uma estratégia a partir da qual todas as futuras decisões podem ser validadas.

Finalidade do sistema

> Descrever a finalidade do sistema >

Escopo do produto

Declaração do escopo do produto

> Seção utilizada para documentar novos produtos. Forneça a declaração do escopo do produto, descrevendo as características do produto, serviço ou resultado que se deseja obter com a execução do projeto.

Caso esteja documentando um produto já existente incluir o texto "**Não se aplica**"

>

Não faz parte do escopo

> Descreva de forma explícita as características que não fazem parte do produto. Em muitos casos, é mais fácil declarar que certos comportamentos nunca poderão ocorrer. Exemplo: O sistema não fará controle financeiro; O sistema não fará estatísticas mensais.

Caso esteja documentando um produto já existente incluir o texto **"Não se aplica"**

Visão geral do produto

Modelagem de processos do negócio

> Cole aqui um diagrama de atividade para representar os processos de negócio (conjunto de atividades que ocorrem em algum negócio com o objetivo de gerar um produto ou serviço, alcançando determinado objetivo). Esse diagrama fornece o entendimento de como são realizadas as diversas atividades contidas em cada processo. >

Descrição

Para descrever funcionalidades/requisitos/features há duas opções:

1. Tradicionais requisitos e casos de uso
2. Utilizar Cucumber/Linguagem Gherkin para definir as features

Requisitos

> Descreva os requisitos funcionais e não funcionais do produto. >

Código	Descrição

Casos de Uso

> Listar e descrever resumidamente as funcionalidades que se espera encontrar no produto. Funcionalidades são capacidades que o produto deve ter para atender a uma necessidade de usuário (ator). Cada funcionalidades descreve um serviço percebido pelo usuário e que tipicamente requer entradas para alcançar o resultado desejado. À medida que o modelo de casos de uso for desenvolvido, atualize a descrição para fazer referência aos casos de uso. Cada funcionalidade será descrita mais detalhadamente no modelo de casos de uso. É recomendado ordenar os casos de uso por ordem de decrescente de prioridade. >

Nome	Descrição	Prioridade	Requisitos Relacionados

Diagrama de Casos de Uso

> Defina aqui o diagrama de casos de uso >

Features

Exemplo de uma feature que obedece à sintaxe do Gherkin/Cucumber:

```
#language: pt
Funcionalidade: Gerenciar Workflows
  Para criar fluxos de tramitação de uma requisição
  Como um usuário com permissão para definição de fluxos
  Eu quero selecionar a sequência de papéis para tramitação de acordo com o curso e tipo de requisição

Cenário de Fundo:
  Dado que o usuário está logado

Cenário: Criar workflow
  Dado que possui permissão para criar workflows
  Quando usuário selecionar opção para criar workflow
  E selecionar o grupo-alvo
  E selecionar o tipo de requisição
  E selecionar ordem dos grupos
  Então solicitar salvamento do workflow na base de dados
```

Mas recusar caso haja algum workflow conflitante

Protótipo de Telas

> Defina aqui os protótipos de Telas >

Documento de Oficialização da Demanda

Documento, assinado pelo beneficiário, que explicita a necessidade da contratação em termos do negócio da organização ([ref](#)).

(Novo Software) [Download](#)

(Alteração de Software) [Download](#)

A fim de aprovar a demanda, deve-se considerar os requisitos do Art. 12. da Lei 8.666 ([ref](#)):

1. segurança;
2. funcionalidade e adequação ao interesse público;
3. economia na execução, conservação e operação;
4. possibilidade de emprego de mão-de-obra, materiais, tecnologia e matérias-primas existentes no local para execução, conservação e operação;
5. facilidade na execução, conservação e operação, sem prejuízo da durabilidade da obra ou do serviço;
6. adoção das normas técnicas, de saúde e de segurança do trabalho adequadas; (Redação dada pela Lei nº 8.883, de 1994)
7. impacto ambiental.

Documento de Arquitetura

Neste documento devem ser anexados todos os diagramas existentes.

Diagramas exigidos

- Modelo do domínio (**ver**)

Diagramas recomendados

- Sequência / Atividade

Linguagens e Guias de Estilo

Este documento enumera as linguagens/frameworks que serão aceitas para o desenvolvimento de sistemas institucionais.

- O motivo de limitar a escolha é usar os conhecimentos mais comuns entre os membros da equipe de desenvolvimento a fim de garantir a manutenção dos sistemas a longo prazo.
- Os guias de estilo devem ser estritamente seguidos e passíveis de checagem automática através de qualquer *linter* disponível que esteja em conformidade com o guia.

Sumário

1. [Python](#)
2. [PHP](#)
3. [Javascript](#)
4. [Java](#)

Python

- Guia de Estilo
 - [PEP 8](#)
- Web
 - [Django \(Padrões\)](#)
 - [Flask](#)
- Arquivos Ignorados no repositório
 - [.gitignore](#)

PHP

- Guia de Estilo
 - [PSR-2](#)
- Web
 - [Laravel](#)

Javascript

- Guia de Estilo
 - (ES6) [Airbnb](#)
- Front-End
 - [AngularJS](#)
 - [jQuery](#)
 - [VueJS](#)
- Back-End
 - [ExpressJS](#)
 - [Sails](#)

Java

- Guia de Estilo
 - [Google Style Guide](#)
- Web
 - [Play Framework](#)
- Mobile
 - [Android](#)

Documentação

Sempre programe considerando que a pessoa que vai manter seu código é um psicopata violento que sabe onde você mora.

Sumário

1. [Código](#)
2. [API](#)

Código

Cada método deve possuir um comentário associado explicando sua finalidade, cada parâmetro e o seu retorno.

Além disso, deve ser possível acessar a documentação fora do código através de uma página passível de navegação. Para isso, podem ser usados geradores automáticos de documentação como, por exemplo, o [Sphinx](#) para Python ou o nativo [Javadoc](#) para Java.

Exemplos:

- Javadoc

```
/**
 * Usado para buscar as inscrições válidas em um determinado edital.
 * Ver {@link remocao.ifpe.edu.br}.
 * @param edital Identificador do edital
 * @return Lista de inscrições válidas.
 * @see Inscricao, Edital
 */
```

- Docstring / Epydoc

```
"""
Essa função é usada para gerar um número aleatório.
```

```
@type max: number
@param max: Número máximo que o número aleatório pode atingir
@type:    number
@return:  número aleatório  $0 < x < \text{max}$ 
"""
```

API

No desenvolvimento de uma API, esta também deve possuir uma documentação *live*. Ver documentação [Swagger](#).

Interface do Usuário

Modelo de página para desenvolvimento de Sistemas IFPE

Sumário

1. [Logotipo](#)
 2. [Manual](#)
 3. [Navbar](#)
 4. [Footer](#)
 5. [Exemplo](#)
-

Logotipo



Manual

Manual de Identidade Visual do IFPE

[Download](#)

Página (Estrutura)

Recomenda-se usar **bootstrap** como *framework* para desenvolvimento *front-end*.

Navbar

```
<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <a href="#"></a>
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Início</a></li>
      <li class="dropdown"><a class="dropdown-toggle" data-toggle="dropdown" href="#">Page 1
<span class="caret"></span></a>
        <ul class="dropdown-menu">
          <li><a href="#">Page 1-1</a></li>
          <li><a href="#">Page 1-2</a></li>
          <li><a href="#">Page 1-3</a></li>
        </ul>
      </li>
      <li><a href="#">Page 2</a></li>
    </ul>
    <ul class="nav navbar-nav navbar-right">
      <li><a href="#"><span class="glyphicon glyphicon-log-in"></span> Entrar</a></li>
      <li><a href="#"></span> Sobre</a></li>
    </ul>
  </div>
</nav>
```

Footer

```
<footer class="text-muted">
  <div class="container">
    <p>&copy; 2016 DADT<p>Todos os direitos reservados</p>
  </div>
</footer>
```

Exemplo

[index.html](#)

eMAG - Modelo de Acessibilidade em Governo Eletrônico

Todos os sistemas desenvolvidos para o IFPE devem ser adaptados ao Modelo de Acessibilidade em Governo Eletrônico para garantir a acessibilidade de todos.

O Modelo de Acessibilidade em Governo Eletrônico (eMAG) tem o compromisso de ser o norteador no desenvolvimento e a adaptação de conteúdos digitais do governo federal, garantindo o acesso a todos. As recomendações do eMAG permitem que a implementação da acessibilidade digital seja conduzida de forma padronizada, de fácil implementação, coerente com as necessidades brasileiras e em conformidade com os padrões internacionais.

Recomendamos o uso da ferramenta *Totally* para verificar automaticamente a acessibilidade do sistema desenvolvido.

O manual completo do eMAG pode ser encontrado no [site](#) do Departamento de Governo Eletrônico.

Release Notes

Para cada nova versão do sistema, deve-se criar um tag específica no repositório anexando um documento estruturado como o exemplo a seguir:

Exemplo

O sistema online **BPMN.ifpe** está com sua primeira versão disponível para acesso interno.

Informações

Produto	BPMN.ifpe
Versão	v0.1
Data	01/12/2016
Acesso	Link
Escopo	Interno
Milestone	Funcionalidades Básicas
Instruções	Manual

Adições

- [UC05] Compartilhar Diagrama
- [UC08] Recuperar versão do diagrama

Removidos

- Log do sistema usando [redis](#)

Mudanças

- Entrar no sistema usando credenciais do novo LDAP

Corrigidos

- Permissões inválidas

Relatórios

Relatório

Modelo

Para cada *Release Candidate* de um sistema, é necessário anexar um relatório de testes executados na requisição de implantação.

Produto	Nome do produto
Milestone	Protótipo v1.0
Categoria dos Testes	(Unitários / Interface / etc.)
Data de execução	dd/MM/yyyy hh:mm
# Casos de Teste	250
# Sucessos	239
# Falhas	11
# Bugs corrigidos	2
# Bugs encontrados	0
Detalhes (link)	www.example.com
Comentários	

Ferramentas

Para a automação da criação de relatórios

- Jenkins
- [xUnit](#)



[Coverage Report](#)



[Workspace](#)



[Recent Changes](#)

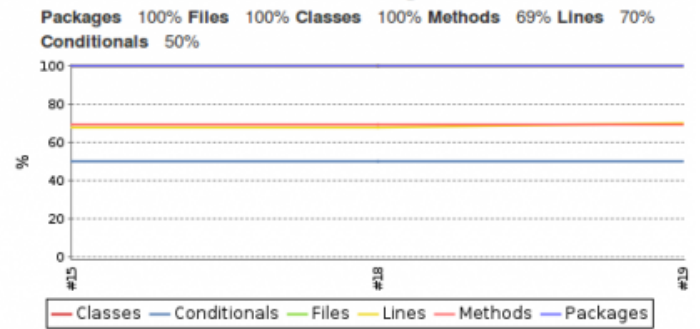


[Latest Test Result](#) (no failures)

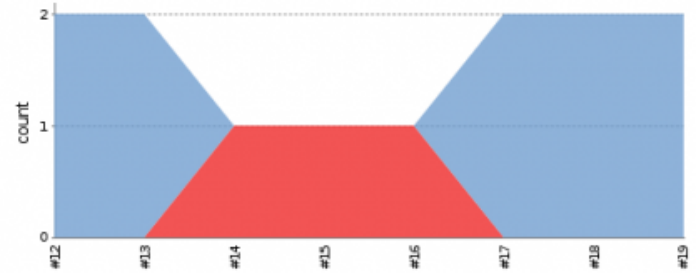
Permalinks

- [Last build \(#19\), 28 min ago](#)
- [Last stable build \(#19\), 28 min ago](#)
- [Last successful build \(#19\), 28 min ago](#)
- [Last failed build \(#17\), 3 hr 7 min ago](#)
- [Last unstable build \(#15\), 3 hr 22 min ago](#)
- [Last unsuccessful build \(#17\), 3 hr 7 min ago](#)
- [Last completed build \(#19\), 28 min ago](#)

Code Coverage



Test Result Trend



Implantação (Web)

A fim de facilitar a implantação do sistema em modo produção, o repositório deve conter os arquivos de configuração necessários para permitir o *deploy* de forma distribuída na infra-estrutura usando [Docker](#) .

Build

A recomendação é que seja criado um `Dockerfile` ([ref](#)) com todas as instruções necessárias para criar uma imagem.

Exemplo:

```
# https://www.digitalocean.com/community/tutorials/docker-explained-how-to-containerize-python-web-applications
FROM ubuntu
MAINTAINER Filipe Arruda (filipe.arruda@reitoria.ifpe.edu.br)

RUN apt-get update
RUN apt-get install -y tar git curl nano wget dialog net-tools build-essential
RUN apt-get install -y python python-dev python-distribute python-pip

COPY /minha_aplicacao /minha_aplicacao

RUN pip install -r /minha_aplicacao/requirements.txt

EXPOSE 80

WORKDIR /minha_aplicacao

CMD python server.py
```

Para se conectar com outros serviços, como por exemplo um banco de dados, é recomendado criar um arquivo `docker-compose.yml`:

```
version: '2'
services:
  db:
    image: postgres
    volumes:
      - app_data: /var/lib/postgresql/data
  web:
    build: .
    volumes:
      - ./minha_aplicacao
    ports:
      - "7000:80"
    depends_on:
      - db
```

Vale salientar que é necessário definir os volumes/diretórios que devem ser salvos fora do *container* para permitir backups/restaurações/escalabilidade.