IFSP - Instituto Federal de Educação, Ciência e Tecnologia
Câmpus São Paulo

Prática de Desenvolvimento de Sistemas - PDS

Informações sobre a disciplina de PDS

IFSP - Instituto Federal de Educação, Ciência e Tecnologia
Câmpus São Paulo

Curso Técnico em Informática Integrado ao Ensino Médio

São Paulo - SP - Brasil

2021-05-08

Histórico de Revisões

Revisão	**Data**	**Autores**	**Descrição**
1.0	2021-05-08	Daniel, Domingos, Ivan, Leonardo	Primeira versão

Lista de abreviaturas e siglas

ABNT Associação Brasileira de Normas Técnicas - Citado em 26, 37

AJAX *Asynchronous Javascript and XML* - Javascript e XML assíncrono - Citado em 20

CDN *Content Delivery Network* - Rede de distribuição de conteúdo - Citado em 17

CSS *Cascading Stylesheet* - Folha de Estilo em cascata - Citado em 17, 18, 20

DNS *Domain Name System* - Sistema de nomes de domínio - Citado em 15, 31

GPS *Global Positioning System* - Sistema Posicionamento Global - Citado em 18, 41

HTML *Hypertext Markup Language* - Linguagem de Marcação de Hipertexto - Citado em 17, 18, 20

HTTP *Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto - Citado em 16, 17, 20, 34, 35

HTTPS *Hypertext Transfer Protocol Secure* - Protocolo de Transferência de Hipertexto Seguro - Citado em 6, 17, 20, 21, 33, 34

IFSP Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Citado em 4, 8, 25, 26, 28, 29

IP *Internet Protocol* - Protocolo de Internet - Citado em 15, 20

JSON *JavaScript Object Notation* - Notação de objetos JavaScript - Citado em 19

MPA *Multi-page Application* - Aplicação de múltiplas páginas - Citado em 17, 20, 21

PDF *Portable Document Format* - Formato de Documento Portável - Citado em 23, 29, 36, 42

PDS Prática de desenvolvimento de sistemas - Citado em 8, 14

PoC *Proof of Concept* - Prova de Conceito - Citado em 31

PWA *Progressive Web App* - Aplicações web progressivas - Citado em 17, 21

RSS *Really Simple Syndication / Rich Site Summary* - Distribuição realmente simples / Sumário Rico de Site - Citado em 25

SGBD Sistema Gerenciador de Banco de Dados - Citado em 20

SPA *Single-page Application* - Aplicação de página única - Citado em 17, 19, 20, 21

SPO São Paulo - Utilizado para indicar o Campus São Paulo do IFSP - Citado em 25

SUAP Sistema Unificado de Administração Pública - Citado em 10, 12, 36

TLS *Transport Layer Security* - Segurança na Camada de Transporte - Citado em 6, 33, 34

URL *Uniform Resource Locator* - Localizador uniforme de recurso - Citado em 27

WCAG *Web Content Accessibility Guidelines* - Diretrizes de Acessibilidade para Conteúdo Web - Citado em 17

YAML *YAML Ain't Markup Language* - YAML não é uma linguagem de marcação - Citado em 19

Sumário

1 Introdução

Este documento descreve o funcionamento da disciplina de Prática de desenvolvimento de sistemas (PDS) do quarto ano do curso de informática integrado ao ensino médio do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP).

1.1 Objetivos de aprendizagem

- Integrar conhecimentos e habilidades adquiridas nos anos anteriores por meio do desenvolvimento de um projeto de software;
- Assumir as responsabilidades de um profissional técnico aplicando os princípios e práticas da área de desenvolvimento de software;
- Fazer julgamentos, tomar decisões éticas e sustentar essas decisões durante o projeto;
- Demonstrar o uso eficaz da comunicação escrita por meio de entregáveis do projeto;
- Praticar habilidades interpessoais durante o trabalho em equipe;
- Atender requisitos relacionados as boas práticas de desenvolvimento de software e gerenciamento de projetos;
- Gerenciar conflitos e riscos do projeto.

1.2 Princípios norteadores

- Estimular a autonomia para pesquisa, organização das informações e tomada de decisão;
- Emular o mundo do trabalho e as dinâmicas do trabalho em equipe;
- Liberdade de escolha de metodologias, plataformas e tecnologias, desde que justificadas e previamente aceitas pelos professores da disciplina;
- Oferecer um ambiente seguro para experimentação, erro e aprendizado.

1.3 Visão geral do projeto

A aplicação a ser desenvolvida deve resolver um problema real, pode ser uma aplicação nova ou também uma evolução ou módulo para uma aplicação já existente de código aberto ou que esteja no repositório de controle de versão.

Essa aplicação pode ser desenvolvida em qualquer plataforma ou linguagem de desenvolvimento desde que compatível com o objetivo (ex uma aplicação para controle de lista de compras não pode ser desenvolvida somente para desktop) e os requisitos apresentados neste documento. A aplicação deve atender aos requisitos apresentados no Capítulo 3.

1.4 Papéis e responsabilidades

Na disciplina existem diversos papéis que são assumidos pelos participantes, o entendimento desses papéis permite atingir corretamente os objetivos da disciplina:

- **Estudante** - Deve desenvolver as atividades da disciplina seguindo os preceitos deste documento e orientações passadas pelos professores, colaborando para o sucesso do projeto desenvolvido pela equipe;
- **Equipe** - Segundo (PARKER, 1995):

  > Um grupo de pessoas com alto grau de interdependência está direcionado para a realização de uma meta ou para a conclusão de uma tarefa, cria-se o conceito de EQUIPE. Em outra palavras, membros de uma equipe concordam com uma meta e concordam que a única maneira de alcançar essa meta é trabalhar em conjunto".

  Desta forma, as equipes são compostas por um número definido de estudantes, que tem como objetivo a concretização do trabalho da disciplina.

  Algumas outras definições de equipes e vídeos de apoio podem ser encontrados em: https://dicas.ivanfm.com/equipes

- **Professor** - Tem o papel de orientar e avaliar, buscando atingir os objetivos da disciplina;
- **Cliente** - Os professores assumem o papel de cliente do projeto e devem ser consultados como um cliente real. Ao desenvolver uma aplicação para resolver um problema real e tendo acesso a usuários reais o projeto pode evoluir muito pois passa por diversas visões em relação ao problema. Uma equipe também pode assumir o papel de cliente de outra equipe desde que não entrem em conflito com as decisões dos clientes principais que são os Professores;

- **Banca Examinadora** - O trabalho é apresentado para uma banca que vai avaliar tanto os documentos demonstrando o desenvolvimento como a aplicação em execução. Essa banca é composta pelos professores da disciplina e convidado(s).

1.5 Avaliações

Todos as atividades desenvolvidas durante o projeto são avaliadas pelos professores, prazos de entregas são considerados como datas de Provas / Avaliações tradicionais e portanto devem ser respeitados.

Além da documentação e da execução da aplicação, são avaliados os modelos estáticos e dinâmicos da aplicação, bem com a relação entre modelos, aplicação e objetivos do projeto.

A divisão das atividades desempenhadas por cada elemento da equipe deve ser documentada no trabalho. A avaliação pode ser individualizada, conforme as atividades desempenhadas por cada aluno ao longo do projeto.

Cada equipe também faz avaliações de seus membros, cada participante avalia todos os membros de sua equipe (incluindo ele mesmo). Essas avaliações também poderão ser consideradas pelos professores ao definir a nota individual de cada participante.

1.6 Comunicação entre alunos e professores

Existem diversos canais de comunicação que poderão ser utilizados durante o desenvolvimento da disciplina, conforme a seguir descrito.

- Comunicador do Sistema Unificado de Administração Pública (SUAP) onde os professores muitas vezes enviam comunicados oficiais que precisam de registro;
- Curso definido no Moodle da disciplina onde algumas atividades devem ser entregues e também permite o envio de mensagens;
- E-mail dos alunos para os professores com dúvidas especificas (sempre enviar com cópia para ambos professores e indicar claramente qual a turma/equipe que faz parte, pois os professores tem projetos de diversas turmas e nem sempre os e-mails chegam com o nome correto do aluno);
- Grupo no Telegram que permite a comunicação entre todos participantes da turma, dúvidas genéricas devem ser feitas principalmente por esse canal pois permitem que todos tenham acesso a informação. Nesse grupo são enviados os links para as aulas/conversas síncronas da disciplina;

- Ferramentas de conferencia (Meet, RNP, Teams, Telegram etc) para conversas síncronas.

É importante lembrar que algumas ferramentas devem ser utilizadas com cuidado, não se deve enviar mensagem com notificação de madrugada por exemplo. No Telegram é possível agendar o envio de uma mensagem ou até enviar sem a notificação, bastando escolher isso no momento do envio.

Existe um canal genérico de projetos (IFSP-SPO-Projetos) onde algumas informações gerais que atendem alunos do ensino médio e superior são publicadas <https://t.me/joinchat/AAAAAET-oEt6v2nyhgx2CQ>.

2 Fases da disciplina

Neste capítulo são apresentadas as fases e tarefas do projeto. Todas as atividades são avaliadas pelos professores durante o período da disciplina e não somente nas entregas. As datas de entregas podem ser encontradas no plano de aulas disponível no SUAP.

2.1 Primeira fase

A primeira fase consiste nas etapas de definição da equipe, análise dos projetos e repositórios, elaboração a apresentação de uma proposta inicial de projeto.

- Definição da equipe (seção 5.1);
- Definição do(a) gerente da equipe (seção 5.3);
- Linha de definição dos acessos no mesmo modelo do acessos.txt do repositório subversion (seção 5.4);
- Criação de blog (seção 5.6);
- Criação de canal no YouTube (seção 5.5);
- Arquivo equipe.yaml validado com yamlint com as definições e urls do projeto, blog, YouTube etc (seção 5.7);
- Pesquisa e avaliação de projetos anteriores (seção 5.9);
- Planejamento (seção 5.2);
- Proposta inicial (seção 5.15);
- Inicio do desenvolvimento do projeto.

2.2 Segunda fase

A segunda fase consiste na entrega da prova de conceito e apresentação do desenvolvimento até o momento.

- Prova de conceito (seção 5.21);
- Desenvolvimento do projeto;
- Apresentação parcial (seção 5.22).

2.3 Terceira fase

A terceira fase consiste na entrega e apresentação da primeira versão e análise dos projetos das outras equipes.

- Entrega primeira versão (seção 5.35);
- Apresentação primeira versão (seção 5.36);
- Acompanhamento das apresentações de todas as equipes e criação de Relatório de análise dos projetos das outras equipes (seção 5.43).

2.4 Quarta fase

A quarta fase consiste na verificação dos relatórios de análises dos projetos e sugestões da banca e correções.

- Ajustes e finalização da aplicação desenvolvida;
- Apresentação final do projeto (seção 5.44).

2.5 Continuamente durante o projeto

- Publicação semanal no blog (seção 5.24);
- A cada apresentação:
  - Auto avaliação da equipe (seção 5.34);
  - *Slides* devem ser colocados no repositório de controle de versão;
  - Vídeo do gource (seção 5.23).

3 Requisitos das aplicações

Os requisitos das aplicações estabelecem partes dos critérios de avaliação do componente curricular PDS e também servem como uma lista de verificação (checklist), facilitando a escolha de diferentes soluções e desenvolvimento do projeto.

3.1 Requisitos aplicáveis a todos os projetos

Todos projetos devem seguir requisitos gerais que são aplicáveis independente da plataforma utilizada.

O desenvolvimento deve ser feito utilizando pelo menos uma linguagem orientada a objetos utilizando os conceitos de orientação a objetos.

Encorajamos a utilização de soluções prontas para problemas comuns (autenticação, validação, CRUD etc), lembrem que vocês devem manter o foco na resolução do problema que é o objetivo da aplicação e a tecnologia deve auxiliar vocês nesse objetivo.

A aplicação deve ser aderente a legislação aplicável (ex LGPD), além de cuidados com critérios de Privacidade, Segurança e Acessibilidade e Performance.

3.1.1 Internacionalização

Com a globalização e disponibilidade de internet uma aplicação pode ter um alcance mundial, para isso as aplicações devem ser pensadas de forma que possam ser utilizadas em mais de um país, a maioria das linguagens já possuem suporte para isso, seguir os padrões de internacionalização desde o inicio do desenvolvimento permite que facilmente uma aplacação seja adaptada para utilização em diferentes países e linguagens.

A aplicação a ser desenvolvida deve ser acessível na língua portuguesa e mais uma outra língua.

- <https://www.w3.org/International/quicktips/index.pt>;
- <https://developer.android.com/training/basics/supporting-devices/languages.html>.

3.1.2 Disponibilidade na internet

A aplicação deve ser disponibilizada para acesso via internet (exceto no caso de aplicação Desktop, para a qual deverá ser disponibilizada a versão distribuível). Existem diversos provedores que oferecem contas gratuitas (AWS, Azure, Google, Oracle Cloud,

Heroku etc) e também existe o convenio AWS Educate onde os professores podem disponibilizar uma sala com créditos adicionais para uso dos alunos que solicitarem.

As contas AWS Educate devem ser criadas com o e-mail institucional @aluno.ifsp.edu.br a partir do link disponível em https://dicas.ivanfm.com/ifsp. Essas contas permitem a criação de maquinas em um Data Center localizado nos Estados Unidos.

Soluções que ficam disponíveis na internet devem ser acessíveis a partir de um nome (via serviço de *Domain Name System* (DNS)) e não somente pelo endereço *Internet Protocol* (IP). Não existe necessidade de registro de um domínio para isso, pode ser utilizado um serviço de DNS dinâmico ou mesmo um nome oferecido pelo provedor de hospedagem.

O serviço deve ficar disponível na internet para que o desenvolvimento e testes possam acontecer da melhor maneira possível.

3.1.3 Desenvolvimento

Durante todo o desenvolvimento os requisitos devem ser seguidos:

- Execução continua de Análise estática (seção 5.28);
- Seguir o *Coding Convention* da linguagem ou definido (e documentado) pela equipe (seção 5.16);
- Código limpo - Martin (2008);
- Controle de versão subversion, seguindo as regras indicadas no repositório[1];
- Preferencialmente utilizar um processo de Integração contínua;
- Sistema de log (seção 5.30);
- Testes (seção 5.31);
- Testes automatizados (seção 5.32).

3.1.4 Volume de dados para demonstração das funcionalidades

As aplicações devem ser testadas e demonstradas com um volume de dados compatível com seus objetivos e também para demonstrar de forma clara o funcionamento correto. Colocar somente um registro em cada tabela não permite demonstrar funcionamentos básicos de paginação, ordenação, filtros e tempo de acesso.

[1] <https://svn.spo.ifsp.edu.br/viewvc/A6PGP/0-LEIA_ME.txt?view=markup>

3.2 Requisitos por plataforma

Uma plataforma é o ambiente específico no qual a aplicação é executada, podendo ser Desktop, Web, Móvel, Embarcado.

3.2.1 Desktop

Aplicações desktop são softwares utilizados para executar tarefas específicas e podem ser instalados em um único computador. Exemplos de aplicações desktop: Processadores de texto como o Apache OpenOffice Writer e Microsoft Word; Editores de imagem como o Gimp e Adobe Photoshop.

A plataforma desktop deve ser considerada caso a solução tenha elevado nível de interação com hardware e sistema operacional e requisitos altos de segurança e proteção da informação.

O desenvolvimento para esta plataforma deve ser avaliado com cuidado, pois carrega um conjunto de desafios que já foram resolvidos com as aplicações web e móvel. Dentre os desafios estão: a distribuição e atualização do software; suporte e problemas relacionados ao ambiente; compatibilidade com diferentes sistemas operacionais; e requisitos mínimos de hardware.

Requisitos para a plataforma desktop:

- Definir como será realizada a distribuição do software;
- Definir licenciamento do software;
- Garantir compatibilidade com os plataformas e sistemas operacionais definidos nos requisitos não funcionais do projeto;
- Definir os requisitos mínimos de hardware;
- Testar os requisitos mínimos de hardware.

3.2.2 Web

Aplicações web são softwares utilizados através de um navegador podendo ser disponibilizados por um servidor *Hypertext Transfer Protocol* (HTTP) de forma remota ou localmente no dispositivo do usuário. Exemplos de aplicações web: Sistemas acadêmicos como o SUAP; E-commerce como a Amazon; Redes sociais como Facebook e Instagram.

Em geral, as aplicações web são construídas usando o modelo de solicitação e resposta do protocolo HTTP. Neste modelo, o cliente (navegador ou outro software) realiza

uma requisição a um servidor que processa e devolve uma resposta. Em seguida, a resposta é tratada e utilizada pelo cliente.

Para garantir a privacidade neste tipo de aplicação deve ser utilizado o protocolo *Hypertext Transfer Protocol Secure* (HTTPS) que é a versão criptografada do HTTP.

Dentre os tipos de soluções para a plataforma web estão: Serviços Web, *Multi-page Application* (MPA), *Single-page Application* (SPA) e *Progressive Web App* (PWA).

Os Serviços Web, em geral, não entregam conteúdo de apresentação (*Hypertext Markup Language* (HTML), *Cascading Stylesheet* (CSS), entre outros formatos). O foco desse tipo de solução é a exposição de funcionalidades e dados para integração entre diferentes partes do sistema ou entre sistemas diversos. Dessa forma, os requisitos gerais para plataforma web a seguir não incluem os requisitos para os serviços web, apenas para as soluções MPA, SPA e PWA:

- Aplicar as técnicas de layout e conteúdo responsivo[2];
- Atender no mínimo os requisitos de nível A das diretrizes de acessibilidade *Web Content Accessibility Guidelines* (WCAG) disponível em <https://www.a11yproject.com/checklist/>;
- Validar código-fonte de apresentação utilizando os serviços indicados a seguir:
  - Validador HTML disponível em <https://validator.w3.org/>;
  - Validador CSS disponível em <https://jigsaw.w3.org/css-validator//>.
- Realizar testes de usabilidade dos principais pontos de interação do software utilizando ferramentas gratuitas;
- Bibliotecas javascript, CSS e similares de terceiros que são disponibilizadas via *Content Delivery Network* (CDN) não devem ser incluídas diretamente no projeto.

3.2.3 Móvel

O desenvolvimento para plataforma móvel não se limita apenas a smartphones, mas a um conjunto de dispositivos como tablets, relógios inteligentes entre outros. Na plataforma móvel um software é chamado de aplicativo ou *app*. O desenvolvimento de aplicativos pode adotar o modelo nativo, multi-plataforma (*cross-platform*) ou hibrido.

Um aplicativo nativo é aquele desenvolvido de forma exclusiva para um determinado sistema operacional, como o Android ou iOS. Neste modelo, o aplicativo é capaz

[2] <https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Responsive/responsive_design_building_blocks>

de consumir diretamente as funcionalidades e recursos do dispositivo e do sistema operacional, tais como câmeras, bluetooth e *Global Positioning System* (GPS). Linguagens de programação como Kotlin ou Java podem ser utilizadas no desenvolvimento Android. Para o sistema operacional iOS são adotadas as linguagens Swift ou Objective-C.

Os aplicativos multi-plataforma (*cross-platform*) permitem gerar executáveis para diferentes plataformas ou sistemas operacionais a partir de uma única base de código. Neste modelo, apenas um aplicativo é desenvolvido e executado em diferentes sistemas como Android e iOS. Isso ajuda as empresas a alcançar um número maior de usuários e também a diminuir o custo com o desenvolvimento e manutenção de versões do aplicativo para diferentes plataformas. Dentre as ferramentas disponíveis para esse modelo de desenvolvimento estão o Flutter[3] e o React Native[4].

O modelo hibrido consiste um uma mistura de uma solução nativa com uma solução web, escrita com as linguagens HTML, CSS e Javascript dentro de um aplicativo nativo usando um mecanismo de *plugin* como Apache Cordova[5] ou Ionic Capacitor[6]. Esse mecanismo de *plugin* permite o acesso aos recursos nativos das plataformas. No entanto, o grande desafio é conseguir a mesma experiência de interação do usuário que é possível nos modelos nativo ou multi-plataforma.

Requisitos para a plataforma móvel:

- Definir como será realizada a distribuição do software;
- Definir licenciamento do software;
- Garantir compatibilidade com as plataformas e versões dos sistemas operacionais definidos nos requisitos não funcionais do projeto;
- Definir e testar os requisitos mínimos de hardware;
- Atender os padrões de interação e uso de componentes definidos em recomendações de cada sistema operacional:
  - Material Design para Android disponível em <https://material.io/>;
  - Human Interface Guidelines para IOS disponível em <https://developer.apple.com/design/human-interface-guidelines/>.

[3] <https://flutter.dev/>
[4] <https://reactnative.dev/>
[5] <https://cordova.apache.org/>
[6] <https://capacitorjs.com/>

3.2.4 Embarcado

Aplicações embarcadas são aquelas que são executadas em um hardware dedicado e interagem com recursos do meio ambiente ou sensores. Essas aplicações normalmente são executadas em equipamentos dedicados como Arduino, ESP8266, Raspberry Pi etc.

Essas aplicações normalmente são conectadas a um servidor Web que gerencia os dados e combina características de Desktop, Web e Móvel.

3.3 Requisitos por tipo de solução

Tipos de solução são definidos por plataforma (Desktop, Web, Móvel e Embarcado). Esses tipos herdam as características e requisitos da plataforma e também podem conter requisitos específicos.

Dependendo da solução ou tipo de projeto a ser desenvolvido, podem ser necessários um ou mais tipos de aplicação, cada qual com sua responsabilidade. Por exemplo, uma SPA normalmente possui uma aplicação front-end (a aplicação com a qual o usuário irá interagir) e um serviço web, com a qual o front-end se comunica. Dessa forma é importante que as técnicas adequadas sejam usadas no desenvolvimento de cada uma dessas aplicações assim como os requisitos necessários sejam obedecidos.

3.3.1 Serviços web

Um Serviço Web é um software desenvolvido para oferecer interação entre sistemas. Eles podem ser criados para utilização por aplicações terceiras ou para divisão de uma mesma aplicação, tornando o sistema mais organizado e isolando todo o processamento da interface com o usuário.

Uma aplicação desenvolvida com base em serviços pode ter uma interface Desktop, Web ou em dispositivo móvel sem necessitar de grandes mudanças, além de permitir o consumo desses serviços por outras aplicações facilitando a integração.

São considerados requisitos para os serviços web desenvolvidos no decorrer do projeto:

- Documentação de acordo com as práticas relacionadas ao tipo de serviço desenvolvido, devidamente hospedados para consulta online: utilizando o padrão OpenAPI 3 em *JavaScript Object Notation* (JSON) ou *YAML Ain't Markup Language* (YAML) para serviços web REST, WSDL para serviços SOAP, ou ainda observando as boas práticas e recomendações do formato / modelo utilizado;

- Validação pelo serviço de todos os dados de entrada, tratando erros e assumindo a possibilidade de requisições inválidas;
- Controle de acesso por meio de autenticação e se necessário autorização;
- Deve estar disponível para utilização através da internet até o final da disciplina na semana de IFA;
- Deve possuir acesso criptografado utilizando HTTPS (seção 5.25);
- Deve ser acessível utilizando um nome e não por endereço IP (seção 5.20).

3.3.2 Multi-page application (MPA)

Aplicações de múltiplas páginas (em inglês "multi-page application", ou MPA) são software desenvolvidos para utilização através de um navegador. Em geral, são aplicações desenvolvidas utilizando tecnologias web como HTML, CSS e Javascript, linguagens de programação, Sistema Gerenciador de Banco de Dados (SGBD) e distribuídas por um servidor de HTTP.

As MPA adotam um projeto de software clássico no qual todo ou a maior parte do conteúdo é construído pelo servidor e devolvido ao navegador. Em geral, para cada interação dos usuários com a aplicação é gerada uma nova requisição ao servidor e recarregamento de página no cliente.

As desvantagens das MPA são o acoplamento entre responsabilidades de front-end e back-end. Além disso, a aplicação deste modelo dificulta a estratégia de lançamento de um produto para plataforma web e após validação do mesmo, o lançamento como um app mobile. Neste caso, um serviço web teria que ser construído a partir de uma MPA, disponibilizando dessa forma acesso as funcionalidades e dados para o app mobile.

Requisitos para projetos que possuem soluções de múltiplas páginas são os mesmos definidos para plataforma web.

3.3.3 Single-page application (SPA)

Assim como as MPA, as aplicações de página única (em inglês "single-page application", ou SPA) são softwares desenvolvidos para utilização através de um navegador. No entanto, elas possuem como objetivo principal fornecer uma experiência de interação similar a uma aplicação desktop. Este objetivo é alcançado por meio do carregamento de todo o código necessário na primeira interação do usuário e em seguida o conteúdo é carregado dinamicamente, sem a necessidade de recarregamento da página.

Em geral, as requisições de conteúdo são realizadas pelo Javascript utilizando variações da técnica conhecida por *Asynchronous Javascript and XML* (AJAX).

Aplicações de página única devem ser consideradas caso a solução necessite de uma experiência de interação de usuário dinâmica, similar a aplicações Desktop e Mobile. Outro requisito que pode direcionar a escolha de uma SPA é a necessidade de separação das bases de código ou projetos, entre back-end/serviços e front-end, devido a questões de organização da equipe, separação de responsabilidades, tamanho do projeto, segurança ou limitação de acesso.

A escolha de SPA para aplicações web que usam os buscadores como fonte primária de tráfego, tais como lojas virtuais, blogs e redes sociais com conteúdo público, deve ser feita com cautela. O conteúdo nas SPA é carregado de forma dinâmica, desta forma, os robôs que fazem a indexação das páginas nos buscadores podem ter acesso somente ao conteúdo inicial e estático, não encontrando páginas internas. Este cenário pode ser resolvido utilizando uma MPA pura, ou com uma solução híbrida MPA e SPA (exemplo de loja virtual na qual o catálogo de produtos é uma MPA e o carrinho de compras e checkout é uma SPA).

Requisitos para projetos que possuem soluções de página única:

- Requisitos definidos para plataforma web;
- Requisitos de cada uma das aplicações envolvidas na construção da solução de acordo com seus respectivos tipos (serviços web, aplicação back-end, aplicação front-end, por exemplo).

3.3.4 Progressive web app (PWA)

Aplicativos Web Progressivos (em inglês "progressive web app", ou PWA) são aplicativos web que usam APIs e recursos dos navegadores, em conjunto com a estratégia de aprimoramento progressivo (em inglês "progressive enhancement") de forma a oferecer uma experiência de usuário semelhante a um aplicativo móvel nativo.

Para considerar uma aplicação web como PWA a aplicação deve atender os seguintes requisitos:

- Distribuição por meio de rede segura (contextos seguros (HTTPS));
- Utilização de *service workers* para interceptação de requisições e implementação de funcionamento *offline*, *cache* e *push notification*;
- Arquivo de manifesto que define informações como nome, ícone e detalhes para que a aplicação web se pareça com um app mobile nativo;

- Cumprimento dos requisitos de cada uma das aplicações envolvidas na construção da solução de acordo com seus respectivos tipos (serviços web, aplicação back-end, aplicação front-end, por exemplo).

3.3.5 Jogos

O desenvolvimento de jogos também é permitido desde que realmente seja feito um desenvolvimento e não somente a utilização de uma ferramenta de criação de jogos por meio de um criador e de *scripts*. Nesse caso em cada turma o número de projetos de jogos deve ser inferior a 50% do total de projetos da turma.

4 Requisitos de gerenciamento dos projetos

Cada equipe tem liberdade para escolher a metodologia que vai utilizar, mas essas metodologias devem ser seguidas corretamente (ex não tem como utilizar Scrum sem utilizar *sprint*).

O objetivo de uma metodologia de gerenciamento de projetos é, além de organizar o trabalho a ser feito, permitir que haja previsibilidade no que será feito, ou seja, ao surgimento de uma nova demanda ou na ocorrência de um imprevisto (como algum problema com um membro da equipe, ou o aparecimento de um requisito não previsto inicialmente), seja possível que os impactos de tal ocorrência sejam identificados e as devidas medidas para mitigação de seus efeitos sejam tomadas, seja pela reorganização do que precisa ser feito, seja pela determinação de ajustes de escopo do projeto.

Ainda, é possível também que, usando uma determinada metodologia como base, ajustes sejam feitos para que a metodologia se adéque ao dia-a-dia da equipe. Porém é importante que os ajustes feitos sejam devidamente explicados (e documentados) e que tais ajustes não impactem os objetivos na adoção de uma de gerenciamento de projetos.

Independente da metodologia escolhida e dos nomes utilizados em cada uma é importante que exista organização e os artefatos correspondentes de cada metodologia sejam utilizados, como:

- Atas;
- Definição de Papéis e Responsabilidades;
- Cronogramas;
- Registro e acompanhamento de métricas;
- Reuniões.

Todos documentos devem ser sempre entregues em formato *Portable Document Format* (PDF) e gerados a partir do LaTeX.

5 Informações sobre as atividades desenvolvidas

Esse capitulo apresenta as atividades a serem desenvolvidas durante o projeto, as seções seguem a ordem de desenvolvimento indicada. Mas algumas atividades podem acontecer paralelamente a outras, e em alguns casos a preparação para o desenvolvimento da atividade deve ser feito antecipadamente.

5.1 Definição da equipe

Os alunos devem se organizar em equipes de acordo com o número definido pelos professores no primeiro dia de aula. Normalmente as equipes variam entre 4 e 6 participantes de acordo com o número de alunos em cada turma.

5.2 Planejamento do projeto

Para se obter o resultado desejado e chegar ao objetivo é importante um planejamento adequado e um acompanhamento da evolução desse plano para chegar ao objetivo.

5.3 Escolha do(a) gerente da equipe

Cada equipe deve definir um Gerente que deve ser responsável por organizar durante todo o período de desenvolvimento do projeto. O Gerente também é responsável por participar das reuniões de gerencia que podem acontecer durante a disciplina, onde os professores podem passar informações para as equipes.

5.4 Envio da linha correspondente ao acessos.txt

Para a criação do diretório correspondente da equipe no repositório de controle de versão (seção 5.10), cada equipe deve enviar para os professores uma linha de texto simples (por e-mail) no mesmo formato do arquivo **acessos.txt** disponível na raiz do repositório de controle de versão. Os prontuários utilizados nesse arquivo devem ser definidos em letras minúsculas, e o acesso ao repositório sempre deve ser feito com login em letras minúsculas.

A maior parte das atividades da disciplina depende do acesso a este repositório, sendo importante que essa atividade seja cumprida no menor tempo para garantir o acesso de todos ao repositório.

5.5 Criação de canal no YouTube

Deve ser criado um canal no YouTube para publicação de todos os vídeos gerados durante o desenvolvimento e apresentações do projeto.

- todos os vídeos publicados devem ter na descrição e nas tags **IFSP**, **SPO** e **PDS**;
- vídeos do gource devem ser publicados nesse canal seção 5.23.

5.6 Criação de blog

Deve ser criado um blog para a equipe com as seguintes características:

- Deve ter um feed RSS ou Atom, de forma a permitir que as publicações sejam visualizadas na íntegra em ferramentas de consumo destes formatos;
- O modelo utilizado para o blog deve ser tal que permita a visualização completa de cada publicação, sem necessidade de clicar em um "ver mais" ou um ícone para permitir a visualização.

A partir da criação do blog, toda semana deverá ser feita uma publicação de acordo com o descrito na seção 5.24

A lista de blogs de projetos anteriores está disponível em: https://dicas.ivanfm.com/blogs-de-trabalhos, muitos desses blogs possuem relatos e dicas que podem auxiliar no desenvolvimento do seu projeto.

5.7 Criação e atualização do arquivo equipe.yaml

Deve ser criado um arquivo no diretório da equipe, no repositório de controle de versão, com as informações sobre a equipe e o projeto que vai ser desenvolvido. Esse arquivo deve ser criado a partir da definição da equipe (seção 5.1) e deverá ser atualizado se houverem mudanças nas informações (ex a primeira versão do arquivo não possui detalhes sobre o projeto a ser desenvolvido, quando o projeto for aprovado pelos professores então o arquivo deverá ser atualizado).

O arquivo deve seguir o mesmo formato dos arquivos já existentes nos diretórios de equipes de projetos anteriores e ser validado com a ferramenta yamllint <https://github.

com/adrienverge/yamllint> que é utilizada no servidor. Outras ferramentas que possuem o mesmo nome podem ter critérios diferentes de validação então é importante a utilização da ferramenta sugerida.

Esse arquivo é a base para definição da página https://dicas.ivanfm.com/blogs-de-trabalhos, caso a sua equipe não apareça nessa listagem provavelmente o seu arquivo é invalido, e na página existe um link para ver o arquivo combinado que contem as mensagens de erro de validação.

Para agilizar a atualização dos dados devem enviar um e-mail para os professores informando que houve a atualização do arquivo.

5.8 Documentos do projeto

Todos os documentos que serão entregues para avaliação devem ser feitos utilizando o LaTeX de acordo com os padrões da Associação Brasileira de Normas Técnicas (ABNT) e do Guia de Normatização do IFSP. Um modelo de documento está disponível em <https://www.overleaf.com/project/58a3a66af9bb74023ba1bd56> e contem exemplos de diversos itens que normalmente são utilizados nesses documentos. O documento modelo também possui informações sobre procedimentos para revisão dos textos.

No repositório de controle de versão existem documentos de outras equipes que também podem ser utilizados como referência. O documento de modelo também possui informações sobre o desenvolvimento de software e problemas encontrados em projetos anteriores tanto em documentação como em relação ao desenvolvimento.

Na página https://dicas.ivanfm.com/textos existem diversas informações uteis para facilitar a elaboração do seu documento. Cuidado com plágio https://dicas.ivanfm.com/plagio, não faça cópias de outros documentos, escreva com suas palavras e referencie a informação.

A ferramenta overleaf onde o modelo de documento se encontra é uma ferramenta de desenvolvimento de textos LaTeX, que possui algumas limitações na versão gratuita, é importante que os alunos mantenham um ambiente de compilação em seus computadores de forma a não ficarem dependentes dessa ferramenta *online*. Algumas atividades como a criação do documento de diferenças entre versões (seção 5.33) dependem de um ambiente corretamente configurado e que não funciona diretamente no overleaf. Alguns professores já possuem contas com características adicionais da conta gratuita do overleaf e podem compartilhar projetos com mais recursos, mas isso não deve ser considerado como a solução normal de uso da equipe.

5.9 Pesquisa e avaliação de projetos anteriores

Durante a definição do projeto a ser desenvolvido deve ser feita uma pesquisa em projetos anteriores disponíveis no repositório de controle de versão (seção 5.10), tanto do curso técnico integrado (**AXXXX***) como do curso superior (**SXXXXYY***) que se encontram no repositório.

Deverá ser feita Análise e Avaliação de dois projetos anteriores que utilizem preferencialmente o mesmo tipo de tecnologia que será utilizada no projeto da equipe deseja desenvolver.

- detalhar quais pontos compatíveis levaram a escolha dos dois projetos analisados;
- gerar documento indicando problemas e melhorias;
- verificar os logs de commits do repositório de controle de versão;
- o mesmo projeto não poderá ser utilizado pelas equipes mais de uma vez para análise, portanto devem se organizar para evitar a duplicação;
- indicar o que aprenderam com os projetos, que cuidados devem tomar a partir da leitura dos outros projetos (tanto os dois analisados e documentados como outros que tenham lido antes de fazer a escolha final dos dois projetos);
- fazer análise crítica, não acreditar no que esta só no documento (verificar se o que esta no documento bate com a realidade que pode ser observada pelo blog, *commits*, arquivos do repositório).

5.10 Utilização do repositório de controle de versão

É obrigatório o uso do repositório de controle de versão indicado pelos professores. A *Uniform Resource Locator* (URL) para acesso é disponibilizada no moodle da disciplina. O acesso deve ser feito utilizando como login o prontuário do aluno (spXXXXXX) sempre em letras minusculas. A senha é a mesma que é utilizada para acesso na rede sem fio do Campus e pode ser redefinida no site <https://central.spo.ifsp.edu.br/>. O repositório é um sistema de controle de versão centralizado Subversion (informações adicionais em https://dicas.ivanfm.com/subversion).

O uso correto do repositório permite o desenvolvimento pelos diversos membros da equipe de forma remota mantendo o conteúdo em local de fácil acesso para todos. Os *commits* devem ser frequentes para garantir que todos tenham acesso ao desenvolvimento atual da equipe, devem seguir as regras indicadas no arquivo **txt** na raiz do repositório. Cada equipe também deve manter seu próprio *backup* dos dados do repositório.

Cada *commit* deve ter um comentário descrevendo que que foi alterado, essa informação é muito importante também para criação posterior do vídeo com a ferramenta gource (seção 5.23).

O Código e documentação de projetos anteriores do curso técnico e superior fica disponível para consulta por todos os alunos do Campus São Paulo do IFSP.

A cada entrega/apresentação deve ser feita uma *tag* no repositório indicando a data e tipo de entrega (ex. 2020-02-10-Prova-de-Conceito)[1].

Se o projeto a ser desenvolvido for um desenvolvimento de funções adicionais para projeto de código aberto, será permitida a utilização de um repositório externo público após a avaliação e aprovação dos professores. O repositório oficial do IFSP deverá ser ser sincronizado pelo menos uma vez a cada 15 dias com a versão atualizada do repositório externo.

5.11 Escolha da metodologia de gerenciamento

Cada equipe tem liberdade para escolha das metodologias utilizadas durante o projeto, para o controle do projeto deve ser escolhida uma (ou combinação de metodologias) de forma a gerenciar a evolução das atividades. É importante que a equipe realmente siga a metodologia escolhida e utilize os artefatos da metodologia.

5.12 Escolha da metodologia de desenvolvimento

Cada equipe tem liberdade para escolha das metodologias utilizadas durante o desenvolvimento, para o desenvolvimento do software deve ser escolhida uma (ou combinação de metodologias) de forma executar o desenvolvimento. É importante que a equipe realmente siga a metodologia escolhida e utilize os artefatos da metodologia.

5.13 Estudo de tecnologias

Antes de iniciar o projeto deve ser feito um estudo das tecnologias que podem ser aplicadas no projeto, pequenos programas de teste podem ser desenvolvidos nesse momento para verificar a aderência dessa tecnologia aos objetivos do projeto. Tecnologias já conhecidas reduzem os estudos, mas deve-se tomar cuidado pois existem casos onde mesmo com um novo aprendizado outras tecnologias podem ser mais eficientes para o projeto.

[1] *tag* é uma função disponível em sistemas de controle de versão, não significa criar um diretório com o nome indicado

É importante considerar também os custos de determinadas tecnologias ou plataformas. Se o objetivo do projeto é criar uma solução para uma entidade sem fins lucrativos e que não tenha verbas para manter um sistema no ar, não vai adiantar escolher uma plataforma que tenha um custo mensal. Além disso algumas plataformas tecnologias possuem limites que podem não ser suficientes para o desenvolvimento da aplicação.

O IFSP possui convênios que permitem acesso a pacotes de serviços de forma gratuita e também existem programas gratuitos dos grandes provedores de nuvem (ex AWS Educate). Analisando projetos anteriores (seção 5.9) é possível obter um grande conjunto de ideias sobre tecnologias possíveis de utilização.

5.14 Definição de tecnologias

Após o estudo das tecnologias passiveis de aplicação no projeto a equipe deve fazer suas escolhas e desenhar o projeto de acordo com as tecnologias estudadas. Em alguns casos os provedores de nuvem permitem escolher onde a aplicação será hospedada, Brasil, Europa, América do Norte etc dependendo do tipo de aplicação a localização pode afetar a latência e a experiencia do usuário.

As escolhas devem ser justificadas e compatíveis com os objetivos do projeto.

5.15 Proposta inicial

Cada equipe deve elaborar uma proposta indicando a aplicação que deseja desenvolver, a proposta deve ser desenvolvida e apresentada para a turma. Os professores tem que aprovar o tema e o escopo do projeto a ser desenvolvido. É importante que as aulas anteriores sejam utilizadas para apresentar e negociar com os professores as possíveis propostas de forma que, no dia da apresentação, já seja apresentada a versão final acordada com os os professores.

- A ordem de apresentação é definida por sorteio;
- Documento PDF gerado no LaTeX, com aproximadamente 5 páginas de texto contendo contextualização, ideia, objetivos, análise de concorrentes, referências utilizadas a partir da análise de projetos anteriores (seção 5.9) e indicação de tecnologias que pretendem utilizar (seção 5.13);
- *Slides* para apresentação da proposta inicial (seção 5.45);
- Uma apresentação de 10 minutos devera ser feita e será seguida de perguntas/comentários dos professores e alunos da turma;

- Vídeo de 5 minutos no YouTube demonstrando essa proposta (seção 5.5);
- Auto avaliação da equipe (seção 5.34) deve ser entregue no repositório.

5.16 Definição de coding convention

A equipe deve definir o padrão de codificação que ira utilizar e seguir durante todo o desenvolvimento de forma que todo o código siga um padrão único. Existem ferramentas de análise estática que também verificam se o formato do código segue um padrão pré-definido, a utilização dessas ferramentas ajuda a seguir esse requisito de desenvolvimento.

5.17 Modelagem de dados

A modelagem de dados é fundamental para o correto funcionamento de um software. Sendo assim, é importante que a equipe efetue a modelagem correta seguindo o método pertinente ao seu projeto (seja pela utilização de um MER, seja pela utilização de um diagrama de classes, quando utilizado um ORM). Ainda, não basta que seja feita a modelagem, mas que se olhe de forma criteriosa para o modelo gerado e que ele seja validado com dados reais pertinentes à solução proposta.

Uma forma simples de verificar se um determinado modelo de dados atente é fazer exercícios simples de inserção, atualização e exclusão considerando os cenários da aplicação, usando ferramentas simples, como planilhas, e, com base nas informações registradas, tentar "responder" a consultas que serão relevantes para seus usuários (observando os requisitos do projeto), de forma a ver se a informação correta será obtida como resposta (e fazendo os devidos ajustes no modelo caso contrário).

5.18 Codificação da aplicação

Durante o desenvolvimento da aplicação diversos critérios devem ser seguidos para garantir que o seu código seja claro tanto para os programadores como para os computadores, os computadores entendem qualquer código que siga os critérios de sintaxe definidos da linguagem, mas as pessoas podem ter dificuldade, a utilização de um mesmo padrão por toda a equipe facilita (seção 5.16), além da utilização de padrões já existentes de estruturas de dados e de codificação. Isso facilita tanto a manutenção por outras pessoas como também pelo próprio desenvolvedor.

Tenham em mente as seguintes frases[2]:

[2] <https://www.javacodegeeks.com/2012/11/20-kick-ass-programming-quotes.html>

- "Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live." Martin Golding;
- "Any fool can write code that a computer can understand. Good programmers write code that humans can understand." Martin Fowler;
- "Good code is its own best documentation. As youre about to add a comment, ask yourself, 'How can I improve the code so that this comment isnt needed?'" Steve McConnell.

5.19 Criação de ambiente para aplicação

A partir das escolhas feitas pela equipe (seção 5.14) deve ser criado o ambiente no provedor escolhido para a aplicação. O ambiente para desenvolvimento e apresentação pode ser uma versão reduzida do ambiente necessário para a aplicação (ex em produção a aplicação precisa de Balanceador de Carga mas para desenvolvimento isso não é necessário). Além disso o ambiente pode ser criado em localização diferente da ideal para redução de custos. Diferenças entre o ambiente proposto para execução da aplicação e o ambiente utilizado devem ser claramente detalhadas na documentação. Considerando que os serviços gratuitos possuem limites é possível a utilização de servidores distribuídos entre contas dos membros da equipe. Além disso é importante tomar o cuidado com serviços que fazem cobrança após o limite gratuito de forma a não terem despesas desnecessárias.

5.20 Disponibilização da aplicação na internet

Para que a aplicação fique disponível na internet deverá ser utilizado um nome de acesso *hostname* em vez de endereço IP. Isso garante que mesmo na mudança do local de hospedagem o endereço de acesso permanece inalterado. Poderá ser utilizado um serviço de DNS dinâmico (dyndns), endereço oferecido pelo provedor de serviço ou um domínio registrado caso a equipe tenha interesse posterior em manter a aplicação (mas não é necessário o registro de domínio próprio para cumprir os requisitos da disciplina)

5.21 Prova de conceito

Deverá ser desenvolvida uma Prova de Conceito (*Proof of Concept* (PoC)) que deve demonstrar a aderência das tecnologias escolhidas com a aplicação que deve ser desenvolvida. Essa prova de conceito deve demonstrar a comunicação desde o usuário até a base de dados e utilizar de forma simples as tecnologias escolhidas para demonstrar que elas funcionam para o objetivo desejado.

- Deve ficar disponível de forma pública na internet;
- Utilizar os ambientes/plataformas escolhidos para implantação (seção 5.19);
- Comunicação criptografada (seção 5.25);
- Demonstrar o funcionamento da internacionalização (duas línguas), utilizando corretamente as funcionalidades da linguagem a partir de arquivos de recursos com as definições textuais de cada linguagem (não deve ser utilizada tradução de página via Google tradutor);
- Demonstrar que os dados do usuário são enviados para o servidor e armazenados na base de dados;
- Demonstrar que APIs especificas atendem as necessidades desejadas;
- Auto avaliação da equipe (seção 5.34);
- Um vídeo de aproximadamente 3 minutos demonstrando a aderência desta prova de conceito com a aplicação final (seção 5.5);
- Um relatório (máximo de 5 páginas) identificando as tecnologias utilizadas e como elas serão utilizadas no projeto (diagrama de arquitetura);
- Vídeo do gource (seção 5.23);
- Uma apresentação de 15 minutos para professores e alunos;
- *Slides* (seção 5.45);
- A ordem de apresentação é definida por sorteio.

5.22 Apresentação parcial

A apresentação parcial consiste em mostrar a situação atual do projeto para os professores e alunos das outras equipes, é importante a utilização de um roteiro para demonstrar de maneira eficiente a situação em que o desenvolvimento se encontra tanto em termos de aplicação como de documentação.

- Auto avaliação da equipe (seção 5.34);
- Vídeo do gource (seção 5.23);
- Uma apresentação de 20 minutos para professores e alunos;
- *Slides* (seção 5.45);
- A ordem de apresentação é definida por sorteio.

5.23 Vídeo do gource

O video criado com a representação do desenvolvimento no controle de versão deve ser feito com a ferramenta gource e deve considerar os seguintes itens:

- Utilizar opção –***key***;
- Utilizar as opções de legendas (***caption***) para registrar as principais mudanças feitas no repositório;
- Deve ter aproximadamente 1 minuto para cada bimestre de desenvolvimento;
- Alterar os ***userid*** do repositório pelos nomes dos participantes;
- Colocar uma imagem distinta e especifica para cada usuário;
- Deve ser publicado no canal do YouTube (seção 5.5).

A cada entrega de aplicação e no final de cada bimestre deve ser criado um novo vídeo do gource.

5.24 Publicação semanal no blog

O blog tem por objetivo servir como registro das atividades realizadas pela equipe a ser acompanhado pelos docentes. Também é importante como memória da equipe, permitindo a consulta pelos membros da equipe de forma a verificar o que ocorreu em uma determinada semana. Dessa forma, é importante considerar o relato do que realmente ocorreu (inclusive pode-se utilizar a publicação combinada a artefatos da metodologia de gerenciamento de projetos utilizada).

A cada semana deve ser publicado no blog pelo menos uma vez indicando as atividades desenvolvidas por cada elemento da equipe. Deve indicar claramente quem foi a pessoa que fez a publicação.

As publicações semanais no blog devem conter um relatório gerencial de andamento do projeto, com identificação do desempenho e das tarefas individuais (ex. reuniões de *milestones*, *project checkpoint*, *sprint review* etc.).

5.25 Comunicação criptografada utilizando HTTPS/TLS

A comunicação entre os módulos da aplicação e usuário devem sempre ser feitas de forma segura. Para isso deve ser utilizado o protocolo HTTPS ou *Transport Layer Security* (TLS) que permitem a comunicação de forma criptografada. A configuração pode ser feita

utilizando um certificado gratuito como o disponibilizado pelo serviço Let's Encrypt ou também utilizando serviços gratuitos como CloudFare, CloudFront etc.

A configuração desse serviço deve ser avaliada e obter uma nota mínima **A** seção 5.26

5.26 Avaliação da configuração HTTPS/TLS

É importante garantir a confidencialidade entre o usuário e a aplicação no servidor, para isso deve ser utilizado um protocolo de criptografia, o mais utilizado é o HTTPS que pode ser configurado com um certificado gratuito como Let's Encrypt, AWS ou similar ou através de um serviço que já ofereça esse acesso criptografado (CloudFront, Cloudflare etc). O processo de criptografia deverá ser documentado e a configuração deverá ser analisada utilizando a ferramenta <https://www.ssllabs.com/ssltest/> onde deve obter no mínimo a nota A (existem poucos casos onde pode aceita uma nota inferior, mas esses casos devem ser negociados com os professores).

5.27 Avaliação de respostas HTTP

Para maior segurança da aplicação e dos usuários existem diversos cabeçalhos que podem ser configurados nas respostas do protocolo HTTP, para validação desses cabeçalhos deverá ser utilizada a ferramenta de análise disponível em <https://securityheaders.io/>, dependendo do tipo de aplicação as condições podem variar e portanto deverá ter a melhor nota possível nessa ferramenta justificando os resultados na documentação.

5.28 Análise estática

O processo de análise estática permite avaliar o código fonte da aplicação sem necessidade de execução dos programas por meio de testes (seção 5.31). É uma atividade incremental que deve ser executada de forma constante de forma a evitar problemas no código. Desenvolver o código de forma limpa e sem erros conhecidos reduz o retrabalho e facilita para todos os membros da equipe.

5.29 Documentação via OpenAPI

A definição dos pontos de chamadas da aplicação deve ser feita utilizando o padrão OpenAPI (padrão que foi definido a partir da ferramenta Swagger). Com a integração de diversas aplicações via protocolo HTTP foi criado um padrão OpenAPI que permite a definição de forma padronizada de como os "serviços"da aplicação podem ser executados.

Isso facilita também a troca de interfaces da aplicação de uma maneira muito simples e também os testes da aplicação. Apesar de existirem ferramentas especificas para chamadas HTTP a definição utilizando um padrão permite a migração entre essas ferramentas de forma transparente.

A documentação deve ser publicada juntamente com a aplicação e somente os pontos principais indicados no documento.

5.30 Sistema de log

A aplicação deverá ser desenvolvida utilizando um ou mais sistema de Log de forma a organizar o registro da aplicação e permitir a configuração do armazenamento de acordo com a plataforma de implantação. A utilização de uma biblioteca já existente facilita no código e também para o administrador. Existem diversas bibliotecas que podem ser utilizadas como: commons-logging, log4j, slf4j, log4php, log4net, logcat etc.

5.31 Testes

Testes garantem que a aplicação atenda os requisitos definidos durante o seu desenvolvimento. É importante que os testes sejam executados repetidamente e de forma padronizada para isso é necessário que seja criado um Plano de testes para execução manual e também a criação de testes automatizados de forma a reduzir a carga de trabalho e evitar erros. Os testes devem considerar tanto os casos de sucesso como de erro, não adianta testar somente as condições principais de uso, pois durante o uso normal dados errados serão colocados pelo usuário e essas condições deverão ser verificadas também.

5.32 Testes automatizados

Os testes automatizados evitam o trabalho manual repetitivo e garantem a maior confiabilidade do sistema. Qualquer um pode e deve executar esses testes sempre que uma nova versão for gerada garantindo que a aplicação esteja se comportando da maneira projetada. Esses testes podem iniciar com testes unitários (que verificam um método/função) e ir até testes complexos de integração onde a aplicação é executada em ambiente de execução de forma a garantir que esteja funcionando corretamente. Para isso devem ser utilizadas ferramentas especificas de automatização de testes pois elas já simplificam esse processo.

5.33 Documento com diferenças - latexdiff

De forma a demonstrar facilmente as alterações feitas entre duas versões de um documento, deverá ser utilizada a ferramenta latexdiff para criação de um documento que demonstra essas diferenças. Cada vez que um documento atualizado for entregue deve ser gerado o latexdiff demonstrando as diferenças para a versão anterior. Por isso é muito importante salvar no repositório de controle de versão a revisão utilizada em cada entrega.

Apesar de existirem algumas versões disponíveis na internet para esse tipo de comparação o documento deve ser gerado localmente em seus computadores, o processo é muito simples mas depende da preparação do ambiente, por isso é importante a preparação antes da data de entrega do documento. O comando recomendado para execução do latexdiff pode ser encontrado em https://dicas.ivanfm.com/latexdiff.

5.34 Planilha de auto avaliação da equipes

Em cada entrega e apresentação cada equipe deverá preencher uma planilha de avaliação onde cada elemento deve dar uma nota, e justificar, para cada elemento da equipe, inclusive a sua própria nota. A planilha de modelo está disponível em https://dicas.ivanfm.com/projetos-e-apresentacoes. Essa planilha deverá ser salva em um arquivo PDF contendo todas as abas originais da planilha e em uma boa formatação para fácil leitura e análise.

5.35 Entrega da primeira versão do projeto

Seguindo o plano de aulas do SUAP deverá ser entregue no repositório de controle de versão a documentação do projeto.

- Documento PDF gerado no LaTeX, com relatório do desenvolvimento (seção 5.37);

5.36 Apresentação da primeira versão do projeto

Cada equipe deve apresentar para a banca examinadora e alunos da turma. Todas equipes devem observar e analisar as apresentações das outras equipes pois devem fazer um relatório avaliando cada outra equipe (Documentos, Apresentação e Aplicação) (seção 5.43)

É importante lembrar que o(s) convidado(s) não participaram do processo vão avaliar o trabalho e portanto sem conhecimento prévio do que é a aplicação ou do que foi o desenvolvimento.

- Slides para apresentação (seção 5.45);

- A apresentação deve ter duração de no máximo 45 minutos, seguida de perguntas dos professores;
- Pelo menos 15 minutos do tempo da apresentação deve ser reservado para demonstração da aplicação desenvolvida (seção 5.42);
- Vídeo de 10 minutos no YouTube demonstrando a aplicação desenvolvida (seção 5.5);
- Vídeo do gource (seção 5.23);
- Auto avaliação da equipe (seção 5.34).

5.37 Relatório do desenvolvimento

O relatório deve seguir o padrão ABNT no formato disponibilizado, deve conter:

- Introdução;
- Problema a ser solucionado;
- Justificativa;
- Objetivos do projeto;
- Revisão da literatura com os assuntos tratados no trabalho:
  - Não misturar o desenvolvimento feito pela equipe com os estudos feitos;
  - Não incluir pesquisa sobre temas das disciplinas do curso, somente temas relacionados ao domínio do problema a ser solucionado;
- Histórico das atividades, incluindo informações sobre a gestão do projeto;
- Reuniões;
- Desenvolvimentos de código, dados;
- Levantamentos;
- Escolhas;
- Descartes (mudanças de rumo durante o projeto);
- Problemas ocorridos no desenvolvimento / gerenciamento;
- Protocolos;
- Modelos de dados, classes, índices de banco de dados etc (seção 5.17);

- Contribuições efetuadas aos projetos de código aberto utilizados;
- Tabela com evolução das métricas do projeto (seção 5.38);
- informações estatísticas do repositório de controle de versão (seção 5.39);
- Links do projeto:
  - Controle de versão;
  - Vídeos;
  - Blog;
  - Aplicação web se for publicada na internet;
  - etc.
- Anexos / Apêndices:
  - Todas as publicações semanais do blog (seção 5.6);
  - Documento de aprovação (seção 5.15)
  - Documento da prova de conceito (seção 5.21)
  - Cronogramas;
  - Planos de teste;
  - Análise de cobertura dos testes;
  - Relatório análise estática (seção 5.28);
  - Manual de usuário (seção 5.40);
  - Manual de técnico (seção 5.41).

Todos os dados qualitativos e quantitativos sobre o projeto (métricas, estatísticas, análises) devem ser efetivamente coerentes com a realidade.

Elementos como as publicações do blog devem ser apresentados na ordem cronológica de publicação.

5.38 Tabela com evolução das métricas do projeto

Para demonstrar a evolução do projeto deverá ser criada uma tabela para demonstrar a evolução das métricas do projeto. Essas métricas devem ser registradas pelo menos uma vez por mês. Caso o projeto tenha módulos distintos (servidor, cliente etc) essa tabela deverá ser separada tendo uma tabela para cada módulo e também tabela com totais. A tabela também poderá ser dividida para facilitar a visualização em tabelas com itens de desenvolvimento, gerenciamento, etc.

Elementos que devem constar:

- Reuniões;
- Publicações de blog;
- Requisitos;
- Tamanho do projeto;
- Arquivos;
- Linhas;
- Classes;
- Interfaces;
- Métodos;
- Atributos;
- Testes (seção 5.31)
- Testes unitários / automatizados (seção 5.32)
  - Classes de testes;
  - Quantidade de testes;
  - Percentual de cobertura;
- Dados sobre análise estática (seção 5.28);
- *commits* (seção 5.10);
- Entidades de banco de dados;
- Imagens;
- Sons;
- Vídeos gerados;
- Outros elementos pertinentes para o projeto ou gerenciamento.

5.39 Informações de utilização do repositório

Para demonstrar a utilização do repositório de controle de versão devem ser apresentadas informações estatísticas geradas pela ferramenta StatSVN. Os dados devem ser comentados. Alguns gráficos são obrigatórios:

- Todos commits porautor https://statsvn.org/statsvn/commitscatterauthors.png;
- Atividade por autor http://statsvn.org/statsvn/activity.png;
- Atividade por dia da semana http://statsvn.org/statsvn/activity_day.png;
- Atividade por hora do dia http://statsvn.org/statsvn/activity_time.png.

Antes de gerar os gráficos os logs do repositório devem ser ajustados de forma a indicar claramente qual foi a pessoa que fez as atividades e não somente o login das pessoas. Dependendo da versão utilizada de subversion existe um bug que já foi detectado e que pode ser contornado facilmente (informações em https://dicas.ivanfm.com/subversion).

5.40 Manual de usuário

Para que o usuário possa utilizar o sistema deverá ser criado um ou mais manuais de usuário (de acordo com possíveis perfis de usuário na aplicação). É bem claro que determinadas aplicações devem ser executadas sem que o usuário tenha que ler um manual, algumas aplicações tem que ter uma interface muito simples para que o usuário não precise ler o manual e para alguns outros casos pequenos vídeos de introdução ou tutoriais dentro da aplicação podem ser suficientes. Existem casos onde é necessário um manual especifico para administrados, moderador etc. A determinação da necessidade do manual escrito ou não deve ser negociada individualmente entre cada equipe e os professores/clientes.

5.41 Manual técnico

O Manual técnico é aquele que permite que desenvolvedores possam fazer manutenções e atualizações no sistema e o administrador de sistemas possa fazer a implantação, configuração, backup e gerenciamento geral. Esse documento pode fazer parte do relatório geral do desenvolvimento ou ser um documento separado. Caso a equipe opte por um manual separado é importante organizar as informações de forma a não ficar repetindo entre os documentos.

5.42 Demonstração da aplicação desenvolvida

Para demonstrar o desenvolvimento a aplicação deve ser apresentada em funcionamento, é importante que seja definido um roteiro para demonstrar as principais funcionalidades e processos da aplicação, a aplicação deve ser populada previamente com dados suficientes para demonstrar o seu funcionamento e entendimento correto, os dados tem que ser compatíveis com o objetivo da aplicação.

Durante uma apresentação com tempo limitado pontos básicos de cadastro devem ser limitados a demonstrar alguns elementos e validações, os processos e funções que a aplicação faz para resolver problemas dos usuários devem ser o foco.

Em alguns casos (como por exemplo utilização de GPS) alguns recursos não podem ser apresentados em tempo real e portando podem ser apresentados por meio de vídeos gravados com antecedência.

5.43 Relatório com análise dos outros projetos

Cada equipe deve assistir as apresentações das outras equipes e analisar o que foi entregue e apresentado gerando um relatório de avaliação indicando pontos negativos, pontos positivos, problemas, criticas, sugestões de melhorias para a entrega seguinte. É importante que cada equipe assista as outras apresentações e já trabalhe no desenvolvimento desse relatório que deve ser entregue ao final do período de apresentações.

O relatório deve ser enviado para o repositório de controle de versão (seção 5.10). Deve ser gerado um documento com uma análise geral de todos os outros projetos e também um documento especifico para cada outra equipe que apresentou. Os documentos devem colocados no repositório de controle de versão com o seguinte padrão:

- **/Documentos/Analise_Outros_Projetos/Resumo.pdf**;
- **/Documentos/Analise_Outros_Projetos/XX_Nome_do_Projeto.pdf**;
- Sendo **XX** a ordem de apresentação (01,02,03 ...) e **Nome_do_Projeto** o nome do projeto que foi analisado.

Cada equipe deverá ler os relatórios das outras equipes e verificar a pertinência, possibilidade e efetuar os ajustes / melhorias além dos pontos apontados pelos professores na banca de apresentação (seção 5.36).

Todos os ajustes devem ser apresentados na entrega final do projeto (seção 5.44).

5.44 Entrega final

Durante a primeira apresentação do projeto (seção 5.36) os membros da banca apresentam suas visões sobre o projeto e devolvem os documentos com anotações e as outras equipes fazem suas análises (seção 5.43). A equipe deve analisar todos as informações e fazer os ajustes necessários para uma nova entrega.

- Apresentação de até 25 minutos das mudanças e correções que foram feitas;
- Relatório do desenvolvimento (seção 5.37) atualizado e documento latexdiff - (seção 5.33);
- Vídeo do gource (seção 5.23);
- Novo vídeo de demonstração da aplicação;
- Auto avaliação da equipe (seção 5.34).

5.45 Slides

Para apresentações devem ser gerados *slides* que tem que ser entregues também no repositório de controle de versão. Esses *slides* devem possuir numeração e podem seguir o modelo que está disponível em <https://www.overleaf.com/project/606fb2f8be5c1d108ec8d237> esse modelo foi desenvolvido também com LaTeX de forma que todos os elementos utilizados na documentação podem ser reaproveitados. Apesar dos *slides* serem um roteiro a seguir durante as apresentações é importante que antes da apresentação seja feito um treino para verificar a sequencia e o tempo da apresentação são compatíveis com o solicitado. Todos *slides* que forem utilizados nas apresentações também devem ser colocados em formato PDF no repositório.

A sua apresentação será vista por pessoas que não participaram do desenvolvimento, é importante a contextualização/introdução do que será apresentado.

Referências

MARTIN, R. C. *Clean Code: A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ, United States: Pearson Education, 2008. (Robert Cecil Martin). ISBN 9780136083252. Disponível em: <https://books.google.com.br/books?id=_i6bDeoCQzsC>. Citado na página 15.

PARKER, G. M. *O poder das equipes*. Rio de Janeiro: Campus, 1995. Citado na página 9.

Glossário

LaTeX	ferramenta que permite gerar documentos sem grandes preocupações com formatação, permite definir funções, macros e outros elementos como um programa - Citado em 44
backup	cópia de segurança de dados - Citado em 27
commit	operação de efetivação de uma alteração, em um banco de dados efetiva um atransação, em um repositório de controle de versão grava um conjunto de alterações que foram efetuadas - Citado em 27, 28, 39
análise estática	ferramenta que analisa o código sem executar de forma a detectar possíveis problemas - Citado em 15, 30, 34, 44
controle de versão	sistema que permite controlar versões de arquivos, normalmente utilizado para versionamento de software, mas qualquer arquivo pode ser controlado. Em arquivos do tipo texto é possível fazer comparação entre versões diferentes. Existem sistemas distribuídos e centralizados - Citado em 15, 44
gource	ferramenta que permite gerar um vídeo a partir dos logs de um repositório de controle de versão - Citado em 13, 25, 28, 32, 33, 37, 42
integração contínua	ferramenta que permite integrar o ambiente de desenvolvimento desde o controle de versão até a implantação nos servidores de teste/produção executando todos os processos definidos incluindo análise estática, testes automatizados e outros - Citado em 15
latexdiff	ferramenta que permite gerar um documento indicando as diferenças entre duas versões de um documento LaTeX - Citado em 36, 42
OpenAPI	Padrão para definição de especificações de API, esse padrão foi definido como uma evolução da ferramenta Swagger - Citado em 6, 19, 34
subversion	sistema de controle de versão centralizado - Citado em 12, 15
testes	procedimentos executados no software de forma a validar se o software está executando de acordo com as definições e requisitos - Citado em 15, 45

testes automatizados — são testes executados de forma automatizada sem intervenção humana, são programados durante o desenvolvimento e executados sempre que o software for alterado de forma a garantir que o software continue executado da forma esperada - Citado em 15, 44