|  |  |
| --- | --- |
| CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC | |
| (  x  ) PRÉ-PROJETO     (     ) PROJETO | ANO/SEMESTRE: 2023/1 |

ASSISTENTE DE ATUALIZAÇÃO DE PROJETOS ANGULAR

Nathan Reikdal Cervieri

Marcel Hugo

# Introdução

[A introdução deve despertar no leitor o interesse pelo texto, apresentando os assuntos que serão tratados e o enfoque que será dado ao tema central. Deve iniciar com uma **contextualização** do estudo a ser realizado, explicando claramente sua origem/motivação. Deve finalizar com a **formulação do problema** (pergunta de pesquisa) a ser investigado.

O tema da pesquisa deve ser abordado de forma clara e sucinta, identificando a situação ou o contexto no qual o problema está inserido. A visão geral do tema deve então ser afunilada até se chegar ao problema a ser pesquisado. Após o problema ter sido identificado, deve-se delimitar que aspectos ou elementos serão tratados. Em resumo, na introdução deve-se deixar bem claro o problema que se quer resolver com o desenvolvimento do trabalho.

**O pré-projeto deve ter no máximo 10 (dez) páginas e o projeto deve ter no máximo 12 (doze) páginas.**

Desenvolvimento de plataformas web e legado

A importância de frameworks para desenvolvimento web

Qual o problema

Delimitação do problema

## OBJETIVOS

O objetivo é simplificar o processo de atualização entre versões do framework Angular para diminuir o tempo e trabalho necessário através da automatização da análise do projeto e alterações a serem feitas para adequar à nova versão.

Os objetivos específicos são:

1. analisar projetos angular;
2. apontar mudanças necessárias para atualização;
3. Realizar mudanças encontradas automaticamente;

# trabalhos correlatos

São apresentados trabalhos com características semelhantes aos principais objetivos do estudo proposto. O primeiro é uma ferramenta de auxílio de atualização de projetos dotnet (DOTNET, 2021), o segundo é um assistente de atualização parecido com o primeiro, mas com foco em atualização de projetos AngularJS para Angular (OLSON, 2018) e o terceiro é uma proposta de método de conversão de projetos legado para tecnologia de nuvem (COSTA, 2018).

## DOTNET UPGRADE ASSISTANT

O Dotnet upgrade Assistant (DOTNET, 2021) é uma ferramenta open source desenvolvida inicialmente pela Microsoft para facilitar o processo de atualização de projetos e soluções dotnet de versões passadas para versões mais atualizadas. Atualmente, suporta os seguintes tipos de projeto: ASP.NET MVC, Windows Forms, Windows Presentation Foundation (WPF); Console app; Libraries; UWP to Windows App SDK (WinUI); Xamarin.Forms to .NET MAUI.

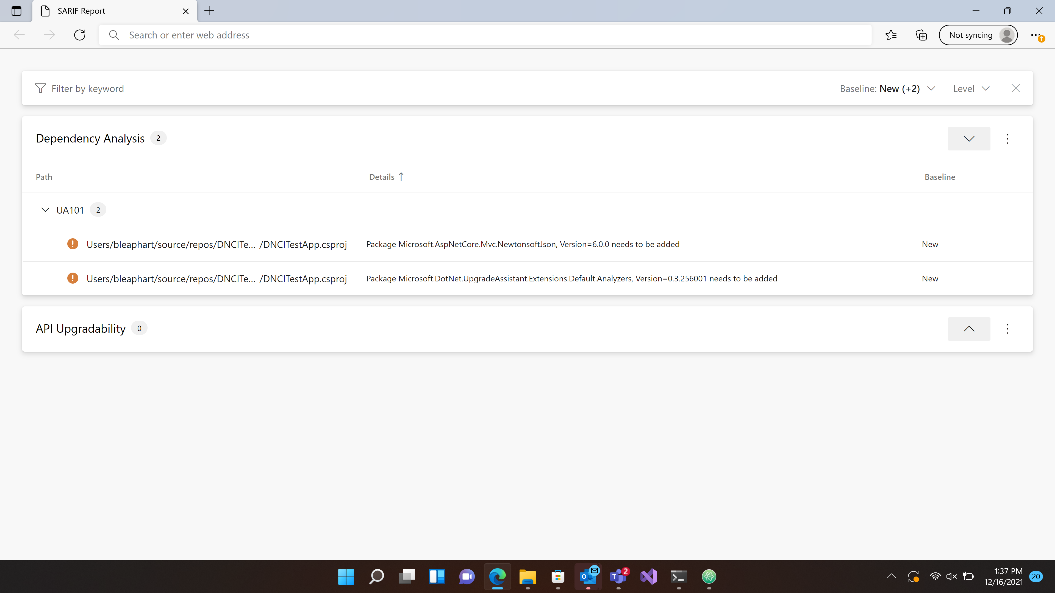
A ferramenta tem 2 pontos principais: Analisar e atualizar. Ao analisar a solução, a ferramenta verifica pacotes, referências, referências a framework e chamadas a pacotes que possam ter alterações que causam quebra e APIs não suportadas, com o intuito de decidir uma ação entre remover, adicionar e atualizar as partes para ficar compatível com a versão de destino.

Ao atualizar o projeto, o Upgrade Assistant determina os projetos e a ordem que eles devem ser atualizados, atualiza o tipo de arquivo, remove pacotes NuGet desnecessários, muda o alvo do pacote para a versão desejada, atualiza os pacotes de acordo com o passo de analise, faz atualizações simples no código C# para garantir compatibilidade, adiciona códigos e partes simples para a versão de destino e adiciona analisadores para ajudar com a atualização.

Após rodar a ferramenta, a aplicação não estará completamente funcional, mas os analisadores criados no passo de update apontam alterações manuais que ainda são necessárias para garantir o funcionamento da solução.

Para utilizar o assistente, se usa uma interface por linha de comando com dois modos, um modo interativo e outro não interativo. Pelo modo interativo, o usuário acompanha a ferramenta durante algumas partes do processo onde tem que decidir como algumas partes serão realizadas, e em outras partes a ferramenta roda de maneira não interativa, realizando alterações de maneira automática. Após todo processo, é gerado um arquivo, ou set de logs que indica o processamento realizado, assim como dar um relatório dos passos como pode ser visto na Figura 1.

Figura - Relatório em HMTL



Fonte: DOTNET (2021).

A ferramenta não é capaz de fazer todos os passos por si só, requerendo interação do usuário em alguns casos, assim como necessita de autenticação para buscar pacotes NuGet relevantes. Atualmente o assistente de atualização não é capaz de analisar a viabilidade e/ou custo da atualização, ao afirmar que se assume que projetos a utilizando já foram revisados.

## NGMIGRATION ASSISTANT

O ngMigration Assistant (OLSON, 2018) é uma ferramenta de linha de comando que escaneia aplicações em AngularJS e recomenda sugestões para realizar a migração para Angular.

A ferramenta analisa o código da aplicação e cria relatórios de anti-padrões de desenvolvimento e quais arquivos e linhas de código necessitam ser alteradas, assim como gera um relatório que indica recomendações e preparações necessárias para realizar a atualização.

Para isso tem alguns passos para realizar a análise: Primeiro busca o caminho de todos os arquivos para serem analisados e quais arquivos para ignorar através de parâmetros de inicialização, depois conta assincronamente as linhas de código encontradas no projeto e executa o passo de análise que realiza vários testes para verificar as alterações necessárias. Finalmente, gera relatórios de anti-padrão e de recomendação para atualização do projeto.

## Uma proposta de migração de sistemas legados do governo para a nuvem

O trabalho de COSTA (2018) é uma tentativa de gerar uma proposta de um modelo de referência para migração de sistemas legados para nuvem, com o intuito de produzir um método de cálculo do Indicador de Percepção de Risco (IPR) cuja validade e aplicabilidade será determinada a partir de um estudo de caso e uma pesquisa de opinião aos órgãos da Administração Pública Federal (APF) que são o foco e os que mais se beneficiarão do artigo.

O artigo retrata o uso de serviços em nuvem pública, os quais apresentam crescimento acelerado. Seu principal benefício é a redução dos investimentos em infraestrutura de TI. De acordo com o artigo é essencial, para o processo de migração para a nuvem, a definição de diretrizes e métodos de migração de sistemas legados que considerem as características do sistema e do provedor que hospedará o sistema.

Primeiramente o trabalho apresenta a evolução dos modelos conceituais de migração para nuvem e critérios para escolha de um modelo a ser referenciado, depois é desenvolvido um método de cálculo para o IPR e um estudo para aplicabilidade, finalmente apresenta uma análise experimental e conclusões assim como trabalhos futuros.

O cálculo definido para o IPR é feito através de quatro passos: Na fase de planejamento e projeto é escolhidos tarefas que representam os pontos mais críticos da migração, depois da escolha se define o peso de cada tarefa em relação as outras tarefas do conjunto, após a seleção destas tarefas e seus pesos as tarefas tem que ser avaliadas em relação ao seu risco em uma escala de um a cinco e finalmente é estabelecido uma classificação de prioridade baseado na percepção de risco avaliada.

Chegou-se à conclusão que, apesar dos riscos e benefícios que a computação em nuvem traz, o modelo apresentado pode gerar muito valor à APF como definido no projeto, apoiado pela amostra de profissionais que analisaram a proposta demonstrando percentuais expressivos, COSTA (2018) também afirma que a proposta adiciona à base de conhecimento em Computação em Nuvem

Para trabalhos futuros evidência a aplicação do método aos sistemas da APF e a criação de um guia para implementação e conversão dos sistemas legados, além de propor a validação do método ao longo da aplicação assim como incorporar alterações do cenário de computação na nuvem ao processo.

# proposta DA FERRAMENTA

Neste capítulo será realizado uma análise dos trabalhos correlatos, assim como uma descrição dos requisitos do trabalho e a exposição da metodologia prevista.

## JUSTIFICATIVA

O Quadro 1 apresenta uma comparação entre os trabalhos correlatos descritos no capítulo 2, expondo suas características e relacionando um aos outros.

Quadro - Comparativo dos trabalhos correlatos

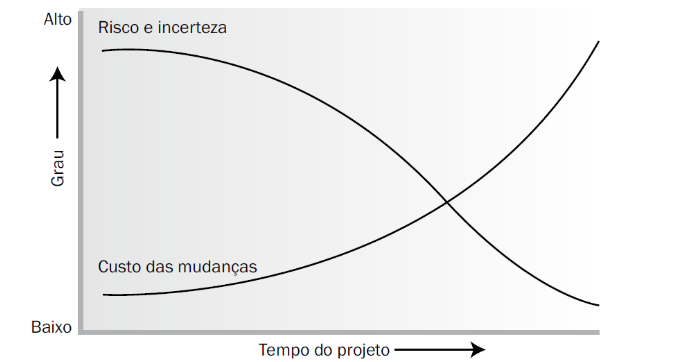
|  |  |  |  |
| --- | --- | --- | --- |
| Trabalhos Correlatos  Características | DONET UPGRADE ASSISTANT | NGMIGRATION ASSISTANT | UMA PROPOSTA DE MIGRAÇÃO DE SISTEMAS LEGADOS DO GOVERNO PARA A NUVEM |
| Migra código | Entre versões e frameworks | Entre frameworks Angular | Entre monolito para nuvem |
| Analisa código | Sim | Sim | Não |
| Sugere alterações necessárias | Sim | Sim | Sim |
| Indica inconsistências entre versões | Sim | Sim | Não |
| Realiza alteração de código | Sim | Não | Não |
| Avalia custo de alterações | Não | Sim, quantidades de linhas a serem alteradas | Sim, IPR |

Fonte: elaborado pelo autor.

Todos os trabalhos correlatos visam a migração de código entre diferentes contextos. O DOTNET (2021) procura fazer essa migração entre frameworks ou versões do mesmo framework, enquanto OLSON (2019) cataloga as alterações necessárias entre dois frameworks do angular e Costa (2018) entre um sistema monolito para infraestrutura de nuvem. Os dois primeiros trabalham diretamente com atualização de código para consistência entre versões já o último lida com infraestrutura. Todos os correlatos fazem sugestões de quais alterações tem que ser feitas para concluir o processo, mas a proposta de migração (COSTA, 2018) não demonstra quais inconsistências podem ser encontradas, e deixa isso para os planejadores. Apenas a ferramenta do DOTNET (2021) faz alteração direta de código, os outros apenas sugerem o que deve ser feito e finalmente, o assistente dotnet (DOTNET, 2021) não avalia qual o custo das alterações necessárias, deixa isso para ser decidido pelos desenvolvedores que irão utilizar a ferramenta, já o assistente de AngularJS (COSTA, 2018) avalia o custo em quantidade de linhas e a migração de sistemas legados (COSTA, 2018) avalia em termo de IPR.

A existência de ferramentas e métodos para agilizar a atualização de código legado e desatualizado é importante para garantir a longevidade de projetos, que com o tempo tem sua qualidade degradada devido a vários desenvolvedores e padrões sendo aplicados, diminuindo sua qualidade e aumentando o custo de desenvolvimento, porém, com menos riscos, considerando que a estabilidade do sistema aumenta com o tempo do projeto, como pode ser visto na Figura 2.

Figura - Custo de mudança ao longo do tempo



Fonte: PMBOK (2017)

A ferramenta proposta visa auxiliar em mudanças de sistemas já antigos feitos no framework Angular para Javascript e HTML, com o intuito de diminuir custo e facilitar a manutenção. O trabalho vai contribuir de maneira científica para catalogar o processo de criação de uma ferramenta deste tipo e dessa maneira tornar o conhecimento mais acessível para desenvolvedores com necessidade de manter sistemas com uma base de código propensa a erros.

## REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

* Permitir análise de projeto completo (RF);
* Permitir análise de arquivos singulares (RF);
* Selecionar a versão de origem e versão objetivo (RF);
* Criar relatório de alterações manuais e automáticas (RF);
* Alterar arquivos de maneira automatizada (RF);
* Criar aplicação em Angular para interface de usuário (RNF);
* Criar servidor C# para analise e substituição (RNF);
* Utilizar banco de dados não relacional para guardar alterações entre versões (RNF);
* Utilizar regex para buscar pontos de alteração (RNF);
* Disponibilizar projeto online (RNF).

## METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

1. Definir método de catalogação: definir objeto de controle de etapa e maneira de guardar esses dados;
2. Catalogar alterações necessárias: utilizar o site que lista alterações em updates do Angular para catalogar as alterações necessárias entre versões;
3. Desenvolver interface visual simples: desenvolver uma interface visual em HTML usando framework Angular para importar arquivo e receber resposta;
4. Desenvolver método de análise arquivo: criar lógica que utilize as ações catalogadas e busque código incompatível nos arquivos utilizando regex;
5. Criar lógica para substituir termos no arquivo: criar lógica que utilize as ações catalogadas e substitua o código incompatível nos arquivos;
6. Implementar retorno de arquivo: retornar arquivo com alterações;
7. Desenvolver relatório de alterações: retornar lista de atualizações realizadas;
8. Criar relatório de alterações manuais necessárias: retornar lista de alterações que não podem ser realizadas automaticamente;
9. Desenvolver interface visual mais robusta: permitir importar projetos através da interface e mostrar os relatórios em tela;
10. Desenvolver método de análise de projeto: utilizar o método de importação de arquivo e realizar análise para todo projeto;
11. Publicar projeto para acesso público: publicar o código do projeto e interface para acesso público.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro - Cronograma

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2023 | | | | | | | | | |
|  | Jul. | | Ago. | | Set. | | Out. | | Nov. | |
| etapas / quinzenas | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| Definir método de catalogação |  |  |  |  |  |  |  |  |  |  |
| Catalogar alterações necessárias |  |  |  |  |  |  |  |  |  |  |
| Desenvolver interface visual simples |  |  |  |  |  |  |  |  |  |  |
| Desenvolver método de análise de arquivo |  |  |  |  |  |  |  |  |  |  |
| Criar lógica para substituir termos no arquivo |  |  |  |  |  |  |  |  |  |  |
| Criar retorno de arquivo |  |  |  |  |  |  |  |  |  |  |
| Desenvolver relatório de alterações |  |  |  |  |  |  |  |  |  |  |
| Criar relatório de alterações manuais necessárias |  |  |  |  |  |  |  |  |  |  |
| Desenvolver interface visual mais robusta |  |  |  |  |  |  |  |  |  |  |
| Desenvolver método de análise de projeto |  |  |  |  |  |  |  |  |  |  |
| Publicar projeto para acesso público |  |  |  |  |  |  |  |  |  |  |

Fonte: elaborado pelo autor.

# REVISÃO BIBLIOGRÁFICA

Neste capítulo será discutido os conceitos que fundamentarão o trabalho como um todo, começando pelo Angular, o framework de desenvolvimento web que o trabalho referência (ANGULAR, 2022), uma visão de sistemas legado e o que são, uma análise de desenvolvimento de aplicações para web, e um

O Angular é um framework de desenvolvimento baseada em typescript (ANGULAR, 2022) criado em 2010 que auxilia na construção de projetos web através de um conjunto robusto de ferramentas e um conjunto de funcionalidades importantes para o desenvolvimento de aplicações como: roteamento, injeção de dependência e vinculação de dados bidirecional (MILLER, 2021), permite assim a construção de aplicações de pequeno e grande porte e facilita a criação de projetos escaláveis de forma rápida e consistente ao remover a responsabilidade de criação de componentes básicos e estrutura de comunicação entre esses componentes dos desenvolvedores (ANGULAR, 2022) (MILLER, 2021).

Com o passar do tempo em projetos de código, sistemas ganham a definição de “sistema legado”. Dedeke (2012) define sistema legado como um “agregado de soluções de software, cuja linguagem, padrões, código e tecnologias pertencem a uma geração passada, ou era de inovação”, o que por definição garante que todo sistema que não passa por atualizações contínuas algum dia será um projeto legado. Sistemas legados podem chegar a custar 300 mil dólares por ano para serem mantidos (DEDEKE, 2012 apud. HEDGE, WALL, 2009). O problema de sistema legado não é algo novo para desenvolvimento de sistemas, vinte e oito anos atrás SNEED (1995) já sugeria um planejamento de reescrita de sistemas legado ao afirmar que não é fácil atualizar e migrar sistemas de larga escala sem perturbar o funcionamento de processamento de dados.

Para se entender o a relevância de sistemas legado e o Angular nesses sistemas, é necessário entender a importância do desenvolvimento web, CHELES (2022) demonstra várias vantagens: a economia e eficiência do desenvolvimento, diminuindo a necessidade de empresas construírem um aplicativo móvel, a função multiplataforma inerente do meio, onde vários dispositivos modernos já possuem a funcionalidade de acessar páginas web e finalmente a conveniência, por não ser necessário baixar ou instalar diferente de aplicações tradicionais.

Para um sistema web funcionar, é necessário haver alguma parte que realiza a lógica do sistema, neste caso será utilizado o C#. A Microsoft (2023) define C# como "uma linguagem de programação moderna, orientada a objetos, *type-safe* e orientada a componente", o que significa que o desenvolvedor tem total controle sobre o domínio da programação, e tendo habilidade de reutilizar componentes sem se preocupar com erros que podem ser causados por falta de controle de tipagem de objeto. A linguagem de programação pode ser utilizada para desenvolvimento de aplicações para desktop Windows, Linux e macOS, assim como desenvolvimento de jogos e aplicativos (ALTEXSOFT, 2021), mas o mais relevante para este trabalho é a possibilidade de utilização para serviços de processamento de requisições e sua funcionalidade como serviço de funcionamento contínuo.

Referências

[Só podem ser inseridas nas referências os documentos citados no projeto. Todos os documentos citados obrigatoriamente tem que estar inserido nas referências.

As referências deverão ser apresentadas em ordem alfabética, de acordo com as normas da ABNT. Como padrão, o nome do autor deve ser apresentado da seguinte forma: sobrenome com todas as letras maiúsculas; primeiro nome por extenso com a primeira letra maiúscula e as demais em minúscula; os outros nomes abreviados (letra em maiúscula seguida de ponto).]

<https://github.com/dotnet/upgrade-assistant>

<https://github.com/ellamaolson/ngMigration-Assistant>

https://repositorio.unb.br/bitstream/10482/34321/1/2018\_BrenoGustavoSoaresdaCosta.pdf

<https://update.angular.io/>

PMBOK 6ª edição pagina 595

https://angular.io/guide/what-is-angular 2022 Acesso 16/04/2023

https://www.codecademy.com/resources/blog/what-is-angular/ MILLER (2021)

A. Dedeke, “Improving legacy-system sustainability: A systematic approach,” It Professional, IEEE, vol. 14, no. 1, pp. 38–43, 2012. 6, 46

H. Sneed, “Planning the Reengineering of Legacy Systems,” IEEE Software, vol. 12, no. 1, 1995, pp. 24–34.

<https://uds.com.br/blog/desenvolvimento-de-aplicacoes-web/>

<https://www.altexsoft.com/blog/c-sharp-pros-and-cons/>

https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/