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

ASSISTENTE DE ATUALIZAÇÃO DE VERSÃO DE PROJETOS ANGULAR

Nathan Reikdal Cervieri

Marcel Hugo

1 INTRODUÇÃO

No cenário da internet é impossível determinar o impacto que páginas da web tem na nossa percepção da tecnologia de comunicação mundial. Tudo pode ser encontrado em páginas da web, desde conhecimentos diversos como a Wikipédia, ferramentas de pesquisa como o Google até websites de compra online como o Mercado livre e a Amazon. Websites são nossa interface de comunicação com as várias facetas do mundo online, e com isso temos a multitude de sistemas e projetos de código relevantes a seus domínios.

Para facilitar o desenvolvimento de páginas da web há vários frameworks de desenvolvimento. Um deles é o Angular, um framework orientado a componente que provê várias funcionalidades que auxiliam a criação de websites interativos, dinâmicos e escaláveis com esforço mínimo (ANGULAR, 2022). De acordo com uma pesquisa feita pela plataforma StackOverflow (2022), Angular está entre as cinco ferramentas mais utilizadas em desenvolvimento web, com aproximadamente 20% das quase 60 mil repostas ao formulário já terem usado a tecnologia.

Porém em todos projetos, independente da tecnologia, existe uma certeza: sistemas algum dia viram um "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", e um projeto legado possui vários problemas inerentes a sua condição, como o custo de manutenção, dificuldade de alteração e incerteza em correções.

O Angular, assim como outras ferramentas, funciona em um sistema de versão, a mais recente é a versão 15, onde novas versões trazem novas funcionalidades e correção de inconsistências ou problemas de segurança. Essas atualizações também trazem alterações que as tornam parcialmente incompatíveis com código escrito previamente. Assim, se o desenvolvedor que utiliza o framework não realizar o esforço para atualizar versões do sistema com frequência pode acabar em um estado onde o custo de atualizar todo o projeto para uma nova versão não parece valer a vantagem que as atualizações trazem. Isso é inaceitável em casos onde a falta de atualização causa falhas críticas de segurança, mas ainda pode ser complicado justificar o custo da atualização contra reescrever partes do sistema do início.

A partir destas afirmações, este projeto pretende desenvolver uma ferramenta que auxilia na atualização de aplicações Angular de modo a diminuir o custo e tempo de alteração. Visa assim diminuir a propensão a erros que refatoração de larga escala pode trazer a um projeto através da análise e alteração automática dos pré-requisitos que podem ser encontrados no site que cataloga alterações necessárias entre versões do Angular (ANGULAR, 2023). E desta maneira garante o funcionamento das funções do sistema entre a versão base e a versão alvo.

1.1 OBJETIVOS

O objetivo é simplificar o processo de atualização de código escrito na versão 14 para a 15 do framework Angular para diminuir o tempo e trabalho necessário através da automatização da análise do projeto e alterações a serem feitas.

Os objetivos específicos são:

a) apontar mudanças necessárias para atualização entre versões 14 e 15 do Angular;
b) realizar mudanças encontradas de maneira automática quando possível;
c) orientar o desenvolvedor sobre mudanças apontadas que não foram possíveis de serem realizadas.

2 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 legados para tecnologia de nuvem (COSTA, 2018).

2.1 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.

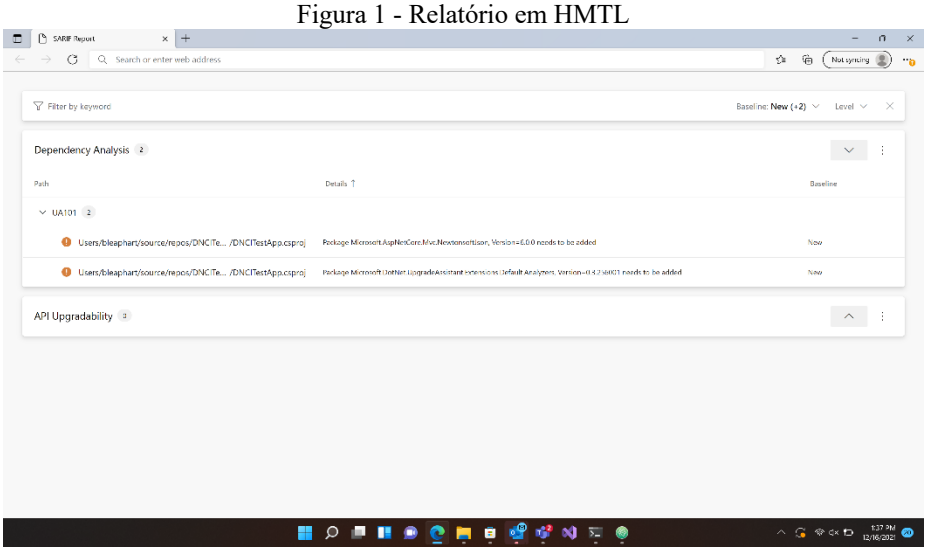
A ferramenta tem 2 pontos principais: Analisar e atualizar. Ao analisar um projeto, a ferramenta verifica pacotes, referências, referências a framework e chamadas a pacotes que possam ter alterações que causam quebra e APIs não suportadas, com o intuito de decidir uma ação entre remover, adicionar e 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 análise, 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 executar 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 em duas partes, uma parte interativa e outra não interativa. Durante a parte interativa, o usuário acompanha a ferramenta durante alguns trechos do processo onde tem que decidir como algumas ações serão realizadas. Em outros trechos a ferramenta roda de maneira não interativa, realizando alterações de modo automático. Após todo processo, é gerado um arquivo, ou conjunto de logs que indica o processamento realizado, assim como gera um relatório dos passos como pode ser visto na Figura 1.

Figura 1 - 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.

2.2 NGMIGRATION ASSISTANT

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

A ferramenta analisa o código da aplicação e cria relatórios de antipadrões 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 antipadrão e de recomendação para atualização do projeto.

O ngMigration Assistant realiza o trabalho de catalogar quais os pontos do projeto onde é necessário realizar algum tipo de alteração para se conseguir atualizar o projeto do AngularJS para Angular, porém não realiza nenhuma alteração de maneira automatizada. O relatório traz também uma contagem das linhas de código que necessitam alteração de maneira a estimar tempo de desenvolvimento, ao apontar que 880 linhas de código é equivalente a 1 mês de trabalho. Conclui-se que a ferramenta é útil para diminuir o trabalho de planejamento de migração, mas não auxilia o processo de atualização em si.

2.3 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. O objetivo é produzir um método de cálculo do Indicador de Percepção de Risco (IPR), cuja validade e aplicabilidade é 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 dos resultados.

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 são escolhidas 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 têm que ser avaliadas em relação ao seu risco em uma escala de um a cinco. E finalmente é estabelecida 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 o trabalho adiciona à base de conhecimento em Computação em Nuvem através de sua proposta.

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.

3 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.

3.1 JUSTIFICATIVA

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

Quadro 1 - Comparativo dos trabalhos correlatos

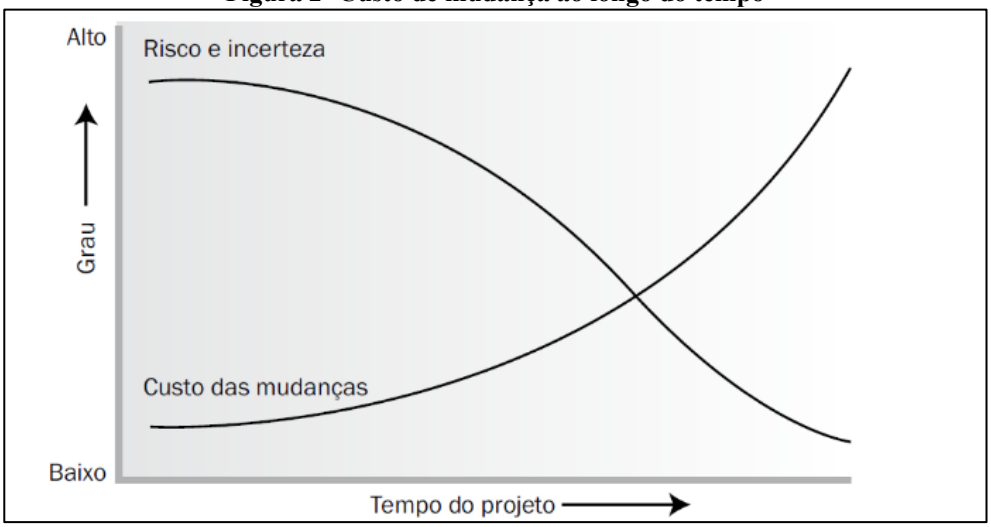
Características \ Trabalhos Correlatos	DONET UPGRADE ASSISTANT	NGMIGRATION ASSISTANT	UMA PROPOSTA DE MIGRAÇÃO DE SISTEMAS LEGADOS DO GOVERNO PARA A NUVEM
Migra código	Entre versões e frameworks	Entre frameworks Angular	Entre sistema legado e arquitetura de nuvem
Analisa código	Sim	Sim	Não
Sugere alterações necessárias	Sim	Sim	Sim
Indica inconsistências entre versões	Sim	Sim	Não
Realiza alteração de código	Sim	Não	Não

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. Apenas a ferramenta do DOTNET (2021) faz alteração direta de código, os outros apenas sugerem o que deve ser feito.

A existência de ferramentas e métodos para agilizar a atualização de código legado e desatualizado é importante para garantir a longevidade de projetos. Com o passar do tempo e manutenções sendo realizadas, sistemas têm sua qualidade degradada devido a vários desenvolvedores e padrões diferentes sendo aplicados. A partir dessa diminuição de qualidade se tem um aumento de 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 2- Custo de mudança ao longo do tempo



Fonte: PMBOK (2017)

A ferramenta proposta visa auxiliar em mudanças de sistemas desenvolvidos na versão 14 do framework Angular com o intuito de diminuir custo e facilitar a manutenção para alcançar a versão 15. O trabalho vai contribuir de maneira científica para catalogar o processo de criação de uma ferramenta deste tipo e também fazer uma análise de literatura sobre o assunto. Dessa maneira também torna o conhecimento mais acessível para desenvolvedores com necessidade de manter sistemas difíceis de atualizar. Também disponibiliza uma ferramenta que possa ser utilizada por desenvolvedores Angular para auxiliar seu processo de atualização entre versões.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A ferramenta deverá:

a) permitir análise de projeto completo (Requisito Funcional - RF);
b) permitir análise de arquivos singulares (RF);
c) criar relatório de alterações manuais e automáticas (RF);
d) alterar arquivos de maneira automatizada (RF);
e) ser desenvolvida em Angular a interface de usuário (Requisito Não Funcional - RNF);
f) ser desenvolvido em C# o servidor de análise e substituição (RNF);
g) utilizar banco de dados não relacional para guardar alterações entre versões (RNF);
h) utilizar regex para buscar pontos de alteração (RNF);
i) disponibilizar projeto online (RNF).

3.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: buscar fontes bibliográficas relacionadas ao objetivo do estudo proposto, como análise de código, alteração automática de código e cálculo de esforço de desenvolvimento;
b) elicitação dos requisitos: reavaliar os requisitos funcionais e não funcionais, com o objetivo de atender o estudo proposto;
c) especificação do trabalho: elaborar diagrama de classes e diagrama de casos de uso;
d) definir método de catalogação: definir objeto de controle de etapa e maneira de guardar esses dados;

e) catalogar alterações necessárias: utilizar o site que lista alterações em updates do Angular para catalogar as alterações necessárias entre versões;
f) desenvolver interface visual simples: desenvolver uma interface visual em HTML usando framework Angular para importar arquivo e receber resposta;
g) desenvolver método de análise arquivo e substituição: criar lógica que utilize as ações catalogadas e busque código incompatível nos arquivos utilizando regex e os substitua;
h) criar relatórios de alterações manuais necessárias e automáticas realizadas: retornar lista de alterações que não podem ser realizadas automaticamente e quais alterações foram feitas;
i) desenvolver interface visual mais robusta: permitir importar projetos através da interface e mostrar os relatórios em tela;
j) teste: buscar ou criar projetos em Angular 14 para testar a funcionalidade esperada na proposta;
k) validação: utilizar os projetos de teste e realizar alterações de maneira manual com intuito de comparar o esforço necessário;
l) 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 2 - Cronograma

	2023									
	Jul.		Ago.		Set.		Out.		Nov.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
levantamento bibliográfico										
elicitação dos requisitos										
especificação do trabalho										
catalogar alterações necessárias										
desenvolver interface visual simples										
desenvolver método de análise arquivo e substituição										
criar relatórios de alterações manuais necessárias e automáticas realizadas										
desenvolver interface visual mais robusta										
teste										
validação										
publicar projeto para acesso público										

Fonte: elaborado pelo autor.

4 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão discutidos os conceitos que fundamentarão o trabalho como um todo, começando pelo Angular (ANGULAR, 2022) e seu guia de atualização de versão; uma visão de sistemas legado e o que são; e um resumo de funcionalidades da linguagem de programação C#.

O Angular é um framework de desenvolvimento baseado em typescript (ANGULAR, 2022) criado em 2010 que auxilia na construção de projetos web através de um conjunto robusto de ferramentas e de funcionalidades importantes para o desenvolvimento de aplicações como: roteamento, injeção de dependência e vinculação de dados bidirecional (MILLER, 2021). Permite 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).

Entre as informações disponíveis para o framework Angular se encontra o Angular Update Guide (ANGULAR, 2023). Essa página da web permite ao usuário indicar quaisquer duas versões do Angular para receber um guia listando as alterações gerais necessárias para adequar da versão mais antiga para mais nova. O guia também permite escolher a complexidade da aplicação, medida em: básica, média e avançada (ANGULAR, 2023). Também é possível escolher entre algumas dependências que interferem na adequação entre versões, sendo elas: o uso de ngUpgrade para combinar AngularJS e Angular, o uso de Angular Material e se o usuário utiliza sistema operacional Windows (ANGULAR, 2023) para assim permitir atualizar projetos que estão em situações diversas.

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 um sistema web funcionar, é necessário haver alguma parte que realiza a lógica do sistema. Neste trabalho 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

ALTEXSOFT. **The Good and the Bad of C# Programming**. 2021. Disponível em: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/. Acesso em: 20 abr. 2023.

ANGULAR. **What is Angular**. 2022. Disponível em: https://angular.io/guide/what-is-angular. Acesso em: 16 abr. 2023.

ANGULAR. **Angular Upgrade Guide**. 2023. Disponível em: https://update.angular.io/. Acesso em: 20 abr. 2023.

COSTA, Breno Gustavo Soares da. **Uma proposta de migração de sistemas legados do governo para a nuvem**. 2018. 110 f. Dissertação (Mestrado) - Curso de Computação Aplicada, Departamento de Ciência da Computação, Universidade de Brasília, Brasília, 2018. Disponível em: https://repositorio.unb.br/handle/10482/34321. Acesso em: 20 abr. 2023.

DEDEKE, Adenekan. Improving Legacy-System Sustainability: a systematic approach. **It Professional**, [S.L.], v. 14, n. 1, p. 38-43, 23 jan. 2012. Bi-Mensal. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/mitp.2012.10.

DOTNET. **.NET Upgrade Assistant**. 2021. Disponível em: https://github.com/dotnet/upgrade-assistant. Acesso em: 20 abr. 2023.

DOTNET. **A tour of the C# language**. 2023. Disponível em: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. Acesso em: 20 abr. 2023.

MILLER, Stephan. **What Is Angular?** 2021. Disponível em: https://www.codecademy.com/resources/blog/what-is-angular/. Acesso em: 20 abr. 2023.

PMBOK. Project Management Institute (ed.). **Um Guia do Conhecimento em Gerenciamento de Projetos**: guia pmbok. 6. ed. Newtown Square, Pensilvânia, Eua: Project Management Institute, 2018. 756 p.

SNEED, Harry M.. Planning the reengineering of legacy systems. **Ieee Software**, [S.L.], v. 12, n. 1, p. 24-34, jan. 1995. Bi-Mensal. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/52.363168.

STACKOVERFLOW. **2022 Developer Survey**. 2022. Disponível em: https://survey.stackoverflow.co/2022/. Acesso em: 20 abr. 2023.