Pontifícia Universidade Católica
do Rio de Janeiro

Rodrigo Galdino Ximenes

Categorização dos tipos de dívida técnica no código de machine learning

Projeto final de programação

Projeto apresentado como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática, do Departamento de Informática da PUC-Rio .

Orientador: Prof. Marcos Kalinowski

Rio de Janeiro
Maio de 2023

Sumário

Lista de figuras

Lista de Abreviaturas

ML – Machine learning

SE – Software engineering

TD – Technical debt

CRISP-DM – Cross-Industry Standard Process Model for Data Mining

AI – Artificial intelligence

GQM – Goal Question Metric

1
Introdução

A metáfora da dívida técnica (DT) foi introduzida por Ward Cunningham em 1992 (CUNNINGHAM, 1992) e é semelhante à dívida financeira, permite mover-se rapidamente na engenharia de software, por outro lado, custa juros compostos a serem pagos posteriormente. Quanto maior o intervalo para limpeza do design e do código, mais difícil será controlar efeitos insatisfatórios, como aumento de prazos para entrega de novas funcionalidades, desmotivação da equipe e outros.

Embora seus efeitos possam trazer benefícios para o projeto, no qual a entrega rápida de funcionalidades ao cliente é o mais desejado, uma análise crítica em termos de gestão e amortização deve ser feita antes de assumir os riscos de acumular DT. Quando se trata de enfrentar DT, sempre há um risco, e é aconselhável cuidar para não fazer mais mal do que bem, pois existe um sistema operacional e a equipe responsável por corrigí-lo deve ser pragmática.

Sculley et al. (SCULLEY et al., 2015) em seu artigo sobre a dívida técnica oculta em sistemas de aprendizado de máquina (ML) que os sistemas ML têm uma capacidade notável de incorrer em DT porque eles possuem todos os problemas de manutenção do código tradicional além de um conjunto adicional de problemas específicos de ML.

O artigo descrito acima lançou luz sobre a relação entre DT e sistemas ML. Como se sabe, DT é um campo tradicional de estudo em engenharia de software (SE), porém seus efeitos e gerenciamento em sistemas de ML ainda são um campo verde para novos estudos.

Com o objetivo de categorizar as possíveis DT presentes em cada etapa do ciclo de vida de um sistema ML, eu, o professor Marcos Kalinowski e a professora Tatiana Escovedo, compilamos uma tabela com os potenciais problemas geradores de DT no código dos sistemas ML e apresentamos para

experientes cientistas de dados da Iniciativa ExACTa da PUC-Rio em duas sessões de Focus Groups no intituito de confirmar e acrescentar informações à tabela mencionada.

Este projeto apresenta uma solução para apoiar os cientistas de dados, possibilitando que eles visualizem as DT presentes em cada etapa do ciclo de vida de um sistema ML, levantados na tabela citada anteriormente e permite que o usuário cadastre novas dívidas ainda não foram mapeadas.

2
Especificação

2.1
Especificação de Requisitos

Esta seção apresenta tanto os requisitos funcionais quanto os não-funcionais do projeto desenvolvido.

2.1.1
Requisitos Funcionais

[RF1] O sistema deve ser capaz de gerar visualização para cada etapa do ciclo de vida de sistemas ML.

[RF2] O sistema deve permitir criar novas DT´S em suas respectivas etapas.

[RF3] O sistema deve permitir editar as DT´S cadastradas.

[RF4] O sistema deve permitir remover as DT´S.

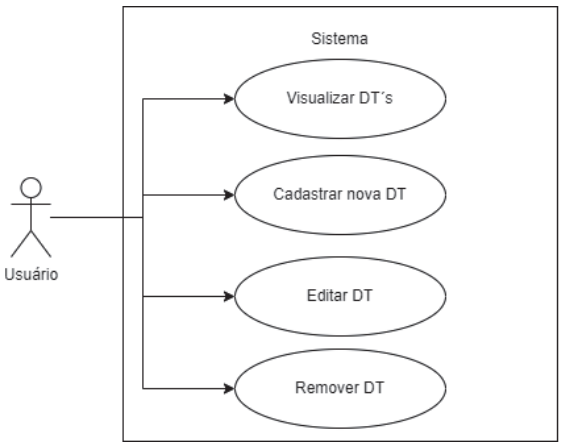

Figura 2.1: Diagrama de Caso de Uso

2.1.2
Requisitos Não-Funcionais

[RNF1] O sistema deve executar em qualquer uma das três distribuições: Windows, Linux e MacOS.

[RNF2] O sistema deve ser possível de ser utilizado após a leitura do Guia do Usuário (passo-a-passo para instalação e inicialização do sistema).

2.1.3 Casos de Uso

O diagrama de caso de uso é mostrado na Figura 2.1

2.1.4 Descrição dos Casos de Uso

Esta subseção descreve os casos de uso encontrados na Figura 2.1. O primeiro caso de uso descreve o relacionamento do usuário com as visualizações pré-estabelecidas, como segue:

UC1 - Visualizar dívidas técnicas

Ator: Usuário

Visão Geral: O usuário abrirá o terminal dentro da pasta *TechDebt* do projeto e digitar o comando *npm install*. Após o comando ser digitado, o usuário deve esperar a conclusão do mesmo para então digitar o comando: *ng serve -o*. Uma aba no navegador irá se abrir e em outro terminal dentro da mesma pasta o comando *json-server - -watch db.json* deve ser digitado.

Trigger: Não existe nenhum.

Pré-Condições: O usuário deve assegurar que sua máquina possui o projeto baixado do git, o node.js, o angular e o angular material instalados.

Pós-Condições: O usuário terá acesso a todas as DT´s já cadastradas no passo selecionado como na Figura 2.2.

Sequência Esperada de Ações:

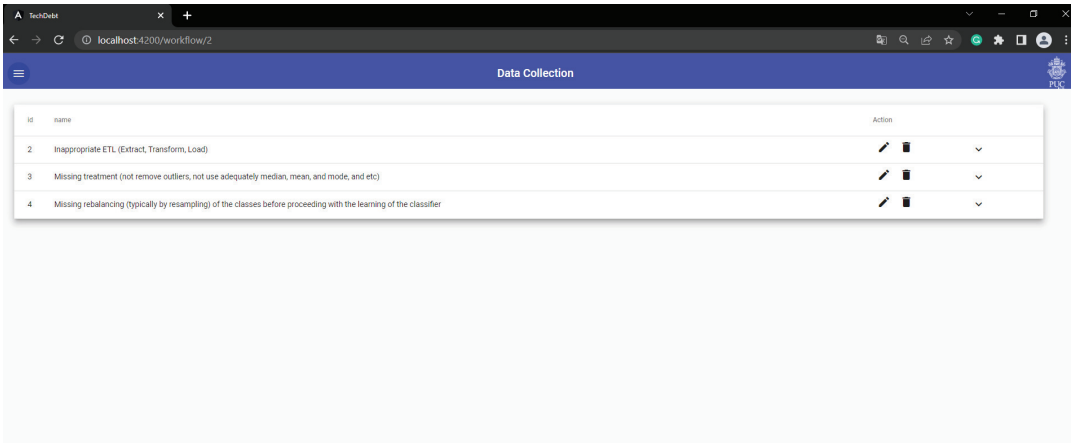

Figura 2.2: Lista de dívidas técnicas

Ações do Ator	Respostas do Sistema
1. Ator digita os comandos no terminal	
	2. O sistema instala as dependências e abre uma aba no navegador padrão do usuário
3. Ator clica no menu e seleciona a etapa do ciclo de vida em questão como na Figura 2.3	
	4. Sistema exibe uma tabela com as DT´s cadastradas

Fluxos Alternativos:

– 2a: O sistema não consegue instalar todas as dependências, o terminal exibe erro e o navegador não abre.

O segundo caso de uso descreve o cadastro das DT´s pelo usuário, como segue:

UC2 - Cadastrar nova dívida técnica

Ator: Usuário

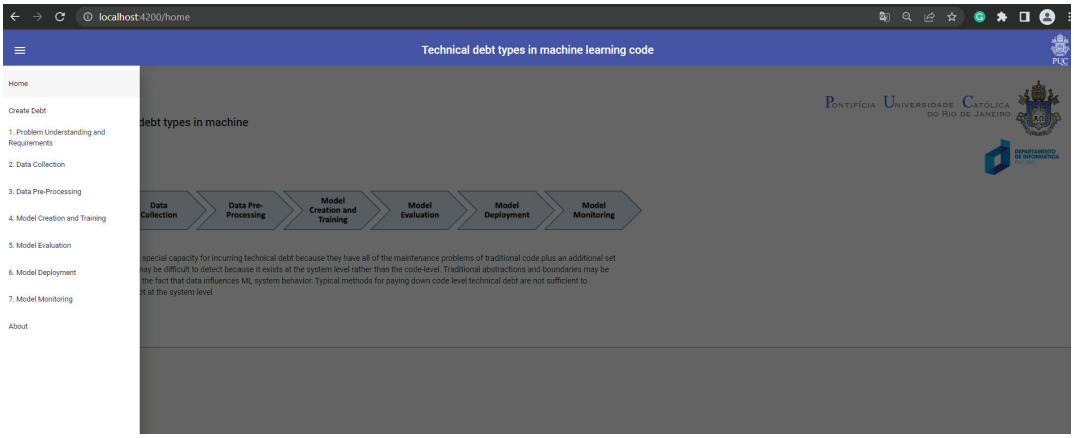

Figura 2.3: Menu inicial

Visão Geral: O usuário abre o terminal dentro da pasta *TechDebt* do projeto e digitar o comando *npm install*. Após o comando ser digitado, o usuário deve esperar a conclusão do mesmo para então digitar o comando: *ng serve -o*. Uma aba no navegador irá se abrir e em outro terminal dentro da mesma pasta o comando *json-server - -watch db.json* deve ser digitado. O usuário irá clicar no menu e ir na opção *Create debt* e a página abrirá como na Figura 2.4.

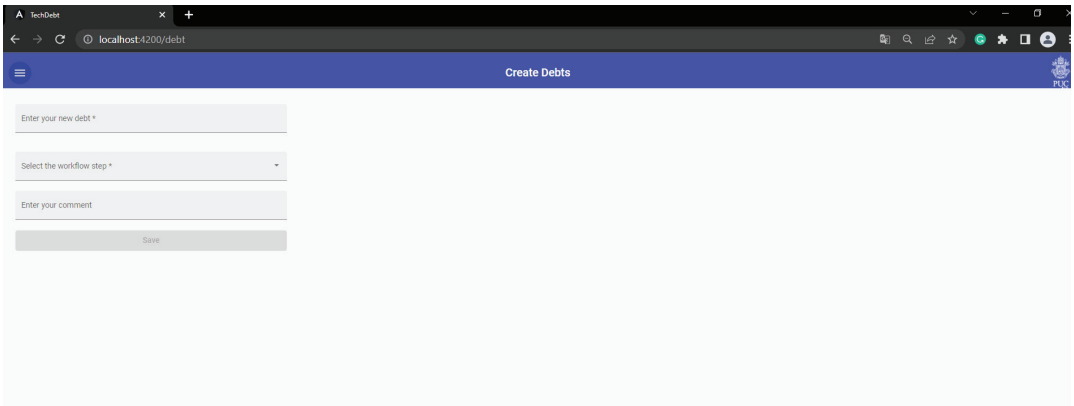

Figura 2.4: Tela cadastro de DT

Trigger: Não existe nenhum.

Pré-Condições: O usuário deve assegurar que sua máquina possui o projeto baixado do git, o node.js, o angular e o angular material instalados.

Pós-Condições: O usuário terá acesso a nova DT cadastrada indo na opção de menu da etapa em que a DT foi inserida.

Sequência Esperada de Ações:

Ações do Ator	Respostas do Sistema
1. Ator digita os comandos no terminal	
	2. O sistema instala as dependências e abre uma aba no navegador padrão do usuário
3. Ator clica no menu e seleciona a opção *Create debt*	
	4. Sistema exibe formulário para preenchimento
5. Usuário preenche formulário e clica em *Save*	
	6. Sistema exibe mensagem *Debt saved* como na Figura 2.5

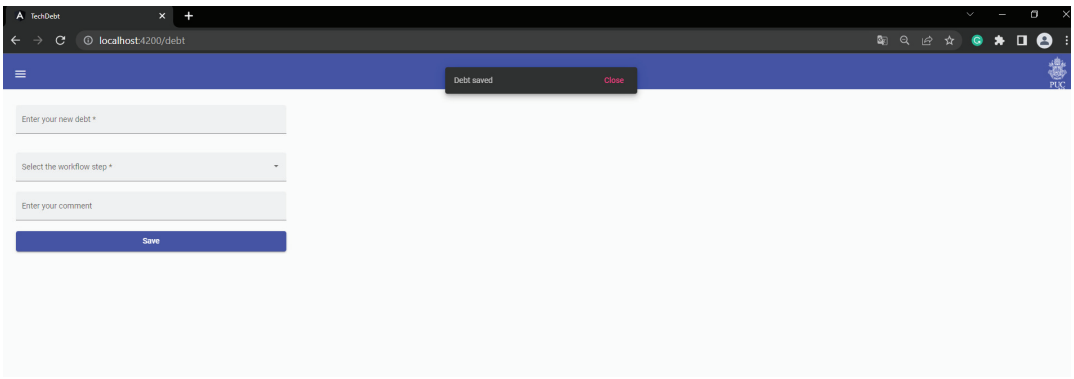

Figura 2.5: Cadastro finalizado

Fluxos Alternativos:

– 2a: O sistema não consegue instalar todas as dependências, o terminal exibe erro e o navegador não abre.

– 5a: O Usuário não preenche de forma adequada os campos do formulário, o sistema envia mensagem de erro para correção do valor preenchido como na Figura 2.6 e desabilita o botão de salvar.

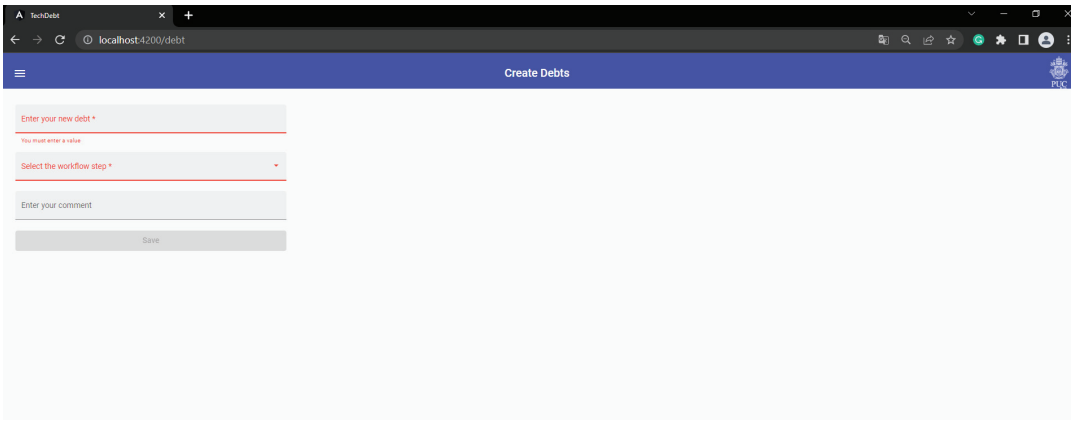

Figura 2.6: Erro no cadastro de DT

UC3 - Editar dívida técnica

Ator: Usuário

Visão Geral: O usuário irá clicar no menu, ir na opção do passo do ciclo de vida que desejar e a página abrirá. Então ele escolherá uma DT e clicará no ícone lápis e a página de edição abrirá.

Trigger: Clicar no ícone lápis na DT selecionada.

Pré-Condições: O sistema deve ter alguma DT previamente cadastrada.

Pós-Condições: O usuário terá acesso a DT editada pois o sistema voltará automaticamente para a tela de visualização das listas de DT´s.

Sequência Esperada de Ações:

Ações do Ator	Respostas do Sistema
1. Usuário clica no menu e seleciona a opção do passo do ciclo de vida que desejar	
	2. Sistema exibe um grid com as DT´s existentes com seus ícones de edição e deleção
3. Usuário clica no ícone lápis de uma linha no grid	
	4. Sistema abre a página de edição com todos os campos preenchidos da DT selecionada
5. Usuário altera os campos necessários e clica em "Salvar"	
	6. Sistema exibe mensagem de sucesso e volta para página de visualização das DT´s

Fluxos Alternativos:

– 5a: O Usuário não preenche de forma adequada os campos do formulário, o sistema envia mensagem de erro para correção do valor preenchido e desabilita o botão de salvar.

UC4 - Remover dívida técnica

Ator: Usuário

Visão Geral: O usuário irá clicar no menu, ir na opção do passo do ciclo de vida que desejar e a página abrirá. Então ele escolherá uma DT e clicará no ícone lixeira e a DT será excluída.

Trigger: Clicar no ícone lixeira na DT selecionada.

Pré-Condições: O sistema deve ter alguma DT previamente cadastrada.

Pós-Condições: O sistema irá automaticamente para a tela de visualização da lista de DT´s.

Sequência Esperada de Ações:

Ações do Ator	Respostas do Sistema
1. Usuário clica no menu e seleciona a opção do passo do ciclo de vida que desejar	
	2. Sistema exibe um grid com as DT´s existentes com seus ícones de edição e deleção
3. Usuário clica no ícone lixeira de uma linha no grid	
	6. Sistema exibe mensagem de sucesso.

3
Projeto do Programa

Para atingir os objetivos do sistema, bem como os requisitos especificados, o sistema foi desenvolvido com framework Angular envolvendo Typescript, CSS e HTML.

3.1
Diagrama de Pacotes

Os recursos desenvolvidos em Angular foram baseados em *Components* e *Services*. A estrutura dos pacotes está disponível na Figura 3.1. Cada *component* criado pelo Angular é composto por um arquivo HTML, um CSS e um Typescript que se relacionam entre si.

Os *services* são responsáveis pela conexão com a API. Eles fazem chamadas para o *JSON Server*, que simula um servidor, respeitando o padrão REST (GET, POST, PUT, DELETE).

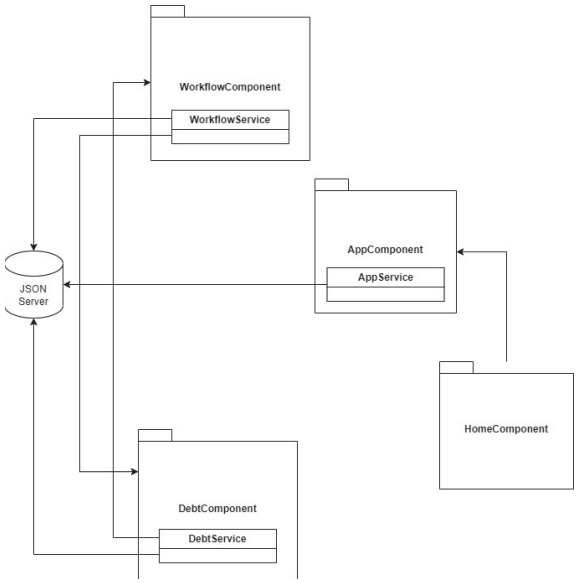

Figura 3.1: Diagrama de Pacotes

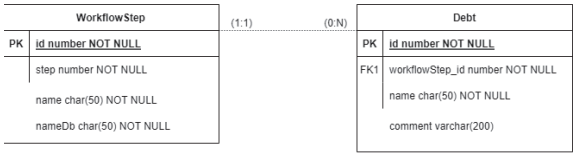

Figura 3.2: Modelagem de dados

3.2 Tecnologias Utilizadas

Por se tratar de um sistema desenvolvido em Angular, que possui diversas bibliotecas disponíveis, é importante listar as que são específicas do projeto. Neste caso:

- Angular Material UI component library: v14.2.6
- JSON Server: v0.17.3
- Angular: v14.2.8
- Node: v14.17.3

3.3 Modelagem de Dados

A modelagem dos dados está repesentada na Figura 3.2. Para modelar todos os potenciais débitos presentes no ciclo de vida de um sistema ML é necessário falar de cada passo especificamente. Então em cada *WorkflowStep* é possível ter nenhum ou até mesmo N *Debts* cadastrados.

4
Código Fonte

O código deste projeto está disponível no ***GitHub***[1], ferramenta usada tanto para acesso dos usuários, quanto para versionamento do mesmo. O código foi escrito em Typescript, HTML e CSS.

O TypeScript é uma linguagem de programação de código aberto que é uma extensão do JavaScript, adicionando tipos estáticos opcionais e outros recursos avançados para ajudar no desenvolvimento de aplicativos mais escaláveis e manuteníveis. É uma linguagem compilada. Isso significa que o código TypeScript é transformado em código JavaScript, que é o código que pode ser executado diretamente nos navegadores ou em ambientes de servidor. Antes de ser executado, o código TypeScript deve ser compilado usando um compilador TypeScript para gerar o código JavaScript correspondente.

[1] https://github.com/rodrigoximenes/ProjetoMestrado

5
Documentação para o Usuário

Nesta seção, alguns pontos sobre a utilidade e utilização do sistema serão respondidas.

5.1
Público-Alvo

O projeto é útil para pesquisadores que estejam interessados em consultar e até mesmo aumentar o catálogo de DT´s de código em sistemas ML.

5.2
Contexto de Atividade

O programa ajuda no entendimento das DT´s em sua etapa no ciclo de vida da criação de sistemas de machine learning.

5.3
Instalação e Execução

A instalação e execução do projeto depende exclusivamente do clone do projeto do *GitHub* e acompanhamento das instruções presentes no README do repositório. De modo resumido, para instalar corretamente o projeto é necessário ter o Node.js. O usuário deve abrir o terminal dentro da pasta *TechDebt* do projeto e digitar o comando *npm install*. Após o comando ser digitado, o usuário deve esperar a conclusão do mesmo para então digitar o comando: *ng serve -o*. Uma aba no navegador irá se abrir e em outro terminal dentro da mesma pasta o comando *json-server db.json* deve ser digitado.

5.4
Contato

Em caso de melhorias e sugestões, na descrição do projeto no *GitHub*, está o meu e-mail para contato.

6
Referências bibliográficas

CUNNINGHAM, W. The wycash portfolio management system. **SIGPLAN OOPS Mess.**, Association for Computing Machinery, New York, NY, USA, v. 4, n. 2, p. 29–30, dec 1992. ISSN 1055-6400. Disponível em: <https://doi.org/10.1145/157710.157715>.

SCULLEY, D. et al. Hidden technical debt in machine learning systems. In: **Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2**. Cambridge, MA, USA: MIT Press, 2015. (NIPS'15), p. 2503–2511.