UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE ENGENHARIA DE SOFTWARE

RELATÓRIO – TRABALHO FINAL QUALIDADE DE SOFTWARE

Neodash

Equipe:

Antônio Hugo Ribeiro Pereira Lobo

Felipe Rodrigues de Santana Freitag

Professora:

Carla Ilane Moreira Bezerra

QUIXADÁ

Outubro, 2023

SUMÁRIO

1 DESCRIÇÃO DO PROJETO

Neodash é uma ferramenta de código aberto, feita utilizando a biblioteca React e utilizando TypeScript, para visualizar dados do Neo4j, que é um sistema de gerenciamento de banco de dados orientado a grafos e é projetado para armazenar, recuperar e manipular dados que possuem relações complexas e interconexões. Além disso, a ferramenta já possui 41 *releases,* mais de 650 *commits* na *branch* principal e 31 *pull requests* esperando análise. Ademais, o projeto está sendo mantido atualmente e possui várias *branches* ativas, tais *branches* buscam adicionar funcionalidades para a aplicação.

O projeto Neodash organiza suas *issues* com alguns rótulos, como: bug, *enhancement*, *documentation*, entre outros. Com isso, eles conseguem uma melhor organização e fazem com que os contribuidores tenham mais facilidade ao procurar uma *issue* para corrigir. Além disso, cada *issue* possui um título e uma descrição para um melhor entendimento do problema e com essa descrição os contribuidores ou até mesmo quem criou a *issue* podem por meio de uma *pull request* corrigir o problema.

Link do projeto: https://github.com/neo4j-labs/neodash

Tabela 1 – Características do Projeto

Projeto	**LOC**	**# de classes**	**# de funções**	**# de *releases***
Neodash	24.317	7	1.633	41

Por se tratar de um código em React o número de classes é baixo, já que em suas versões atuais os componentes são implementados utilizando funções e, por isso, o número de funções está sendo reportado.

2 *ISSUES* DO PROJETO DE SOFTWARE LIVRE

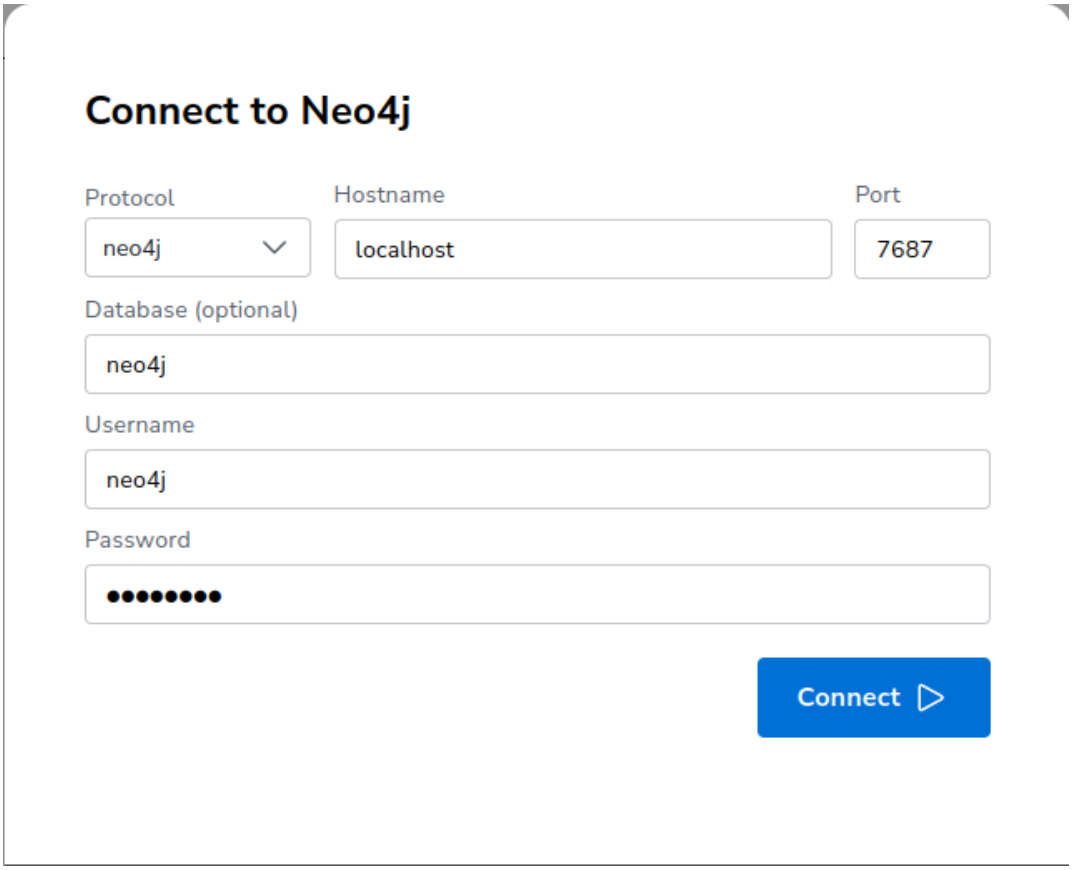
O projeto possui 131 *issues* abertas e os possíveis tipos são: *bug, documentation, duplicate, enhancement, good first issue, help wanted, priority, question, to do* e *wontfix*. O significado delas é:

- ***Bug:*** utilizado para *issues* que são frutos de bugs no código.

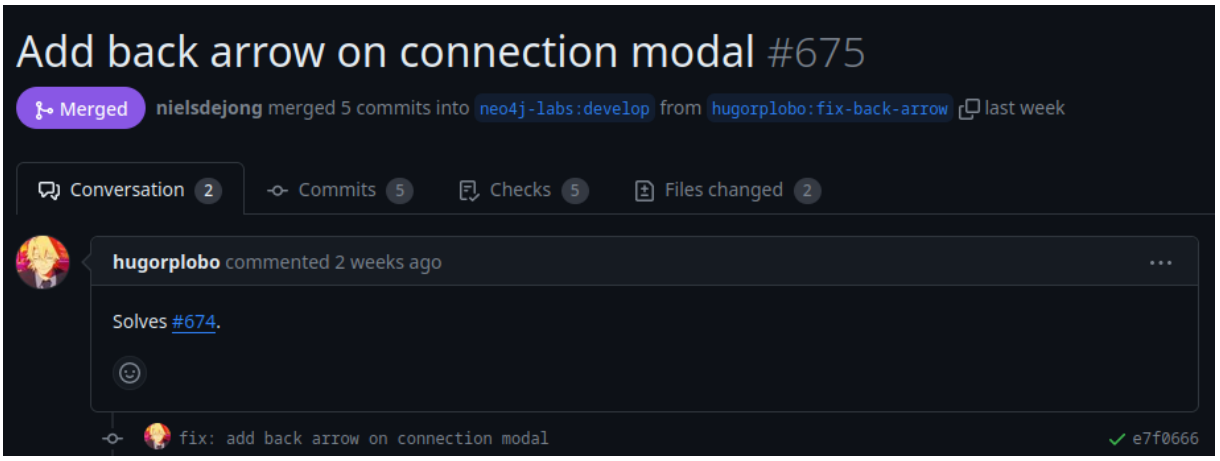
- ***Documentation:*** utilizado para quando se trata de uma melhoria ou adição na documentação.
- ***Duplicate:*** utilizado para quando a *issue* é igual a outra que já está aberta.
- ***Enhancement:*** utilizado para pedidos de melhoria ou adição de novas funcionalidades.
- ***Good first issue:*** *issues* boas para novatos que estão querendo contribuir com o projeto.
- ***Help wanted:*** utilizado para quando os contribuidores precisam de ajuda para resolver a *issue*.
- ***Priority:*** a *issue* deve ser considerada antes das demais.
- ***Question:*** utilizado para quando a *issue* aberta se trata de uma pergunta.
- ***To do:*** foi definido o que precisa ser feito, mas ainda precisa ser implementado.
- ***Wontfix:*** utilizado para *issues* que os contribuidores não pretendem resolver.

De todas as *issues* abertas, nenhuma delas parecia muito fácil de resolver por se tratar de um sistema complexo, então abrimos o sistema e procuramos identificar algum *bug* simples que conseguíssemos resolver. Após algum tempo procurando conseguimos achar um problema relacionado a *user experience*, o problema era seguinte: caso o usuário clicasse em criar um dashboard ou acessar um dashboard sem querer e quisesse voltar, não havia um botão para isso no modal, ou seja, o único jeito dele voltar era atualizando a página e voltando para o menu inicial. A partir disso buscamos entender o código e adicionar um fluxo que corrigisse esse problema. Ademais, para abrir a *issue* tivemos que primeiro selecionar se o que iríamos relatar se tratava de um *bug* ou sugestão de melhoria, após marcamos *bug* fomos apresentados as diretrizes impostas que eram: *issues* devem ser para avisar sobre *bugs* ou sugerir melhorias, o usuário deve procurar nas outras *issues* se já não há algo relacionado, o usuário deve dizer a versão do sistema, entre outras.

Figura 1 - Problema de navegação encontrado



A *issue* pode ser encontrada nesse link, e a *pull request* aberta pode ser vista aqui. Após a revisão por parte dos desenvolvedores do Neodash a *pull request* foi aceita e a *issue* fechada.



3 AVALIAÇÃO DO PROJETO

3.1 Medição 1 – Antes de refatorar o projeto

Utilizando a ferramenta Understand as seguintes medições foram coletadas e apresentadas na tabela abaixo.

Tabela 2 – Medição dos atributos antes de refatorar o projeto.

Sistema	**Complexidade**				**Tamanho**		
	CC	ACC	SCC	MaxNest	LN	CLOC	CountDeclFunction
Métricas antes da refatoração	3.272	1,93	9.594	14	57.043	3.413	1.933

Significado das siglas e termos da Tabela 2: CC - Complexidade Ciclomática, ACC - Complexidade Ciclomática Média, SCC - Complexidade Ciclomática Total, MaxNest - aninhamento máximo, LN - número de linhas físicas, CLOC - número de linhas de comentário, CountDeclFunction - número de declarações de funções.

3.2 Detecção dos *Code Smells*

Os dados coletados abaixo a respeito dos *code smells* foram coletados utilizando a ferramenta Reactsniffer capaz de detectar *code smells* em códigos feitos utilizando a biblioteca React. Além disso, a refatoração dos *smells* pode ser vista nesse fork: https://github.com/hugorplobo/neodash, no qual existem *branches* para cada tipo de *code smell* que foi resolvido.

Tabela 3 – *Code smells* do projeto.

Nome do *Code Smell*	**Quantidade**
Large Component	52
Too many props	47

Any Type	12
Enum Implicit Values	7
Missing Union Type	5
Direct DOM Manipulation	2
Inheritance instead of composition	2
JSX outside the render method	1
Uncontrolled component	1

3.3 Medição 2 - Após refatorar o *smell Enum Implict Values*

Em relação a refatoração desse *smell* foi necessário adicionar valores específicos aos itens do Enum, a correção desse *smell* busca prevenir futuros erros que podem ser ocasionados ao adicionar mais itens ao Enum. Foram refatorados 7 *smells* do tipo *Enum Implicit Values*. A esquerda está uma imagem antes da refatoração e a direita uma imagem após a refatoração.

```
enum Menu {
  DASHBOARD,
  DATABASE,
  CREATE,
  NONE,
}
```

```
enum Menu {
  DASHBOARD = 0,
  DATABASE = 1,
  CREATE = 2,
  NONE = 3,
}
```

Tabela 4 – Medição dos atributos após refatorar o *smell Enum Implicit Values*.

Sistema	**Complexidade**				**Tamanho**		
	CC	ACC	SCC	MaxNest	LN	CLOC	CountDeclFunction
Após a refatoração do *smell Enum Implicit Values*	3.272	1,93	9.594	14	57.043	3.413	1.933

É notório que nenhuma das métricas mudou, pois a maneira como esse *smell* é refatorado não influencia nenhuma delas. A *pull request* que corrige esse code *smell* pode ser vista aqui.

3.4 Medição 3 - Após refatorar o *smell Missing Union Type*

Em relação a refatoração desse *smell* foi necessário a criação de tipos para os Unions do TypeScript. Foram refatorados 5 *smells* do tipo *Missing Union Type*. A primeira imagem é como estava antes da refatoração e a segunda imagem é como ficou após a refatoração.

```
const handleSettingsMenuOpen = (
  event: React.MouseEvent<HTMLElement> | React.KeyboardEvent<HTMLElement>
) => {
  setAnchorEl(event.currentTarget);
};
```

```
type SettingsMenuOpenEvent = React.MouseEvent<HTMLElement> | React.KeyboardEvent<HTMLElement>;

export const NeoDashboardTitle = ({
  dashboardTitle,
  setDashboardTitle,
  editable,
  isStandalone,
  dashboardSettings,
  extensions,
  updateDashboardSetting,
  connection,
}) => {
  const [dashboardTitleText, setDashboardTitleText] = React.useState(dashboardTitle);
  const [anchorEl, setAnchorEl] = useState<HTMLElement | null>(null);
  const [editing, setEditing] = React.useState(false);
  const debouncedDashboardTitleUpdate = useCallback(debounce(setDashboardTitle, 250), []);

  const handleSettingsMenuOpen = (event: SettingsMenuOpenEvent) => {      You, 6 hours ago • refa
    setAnchorEl(event.currentTarget);
  };
```

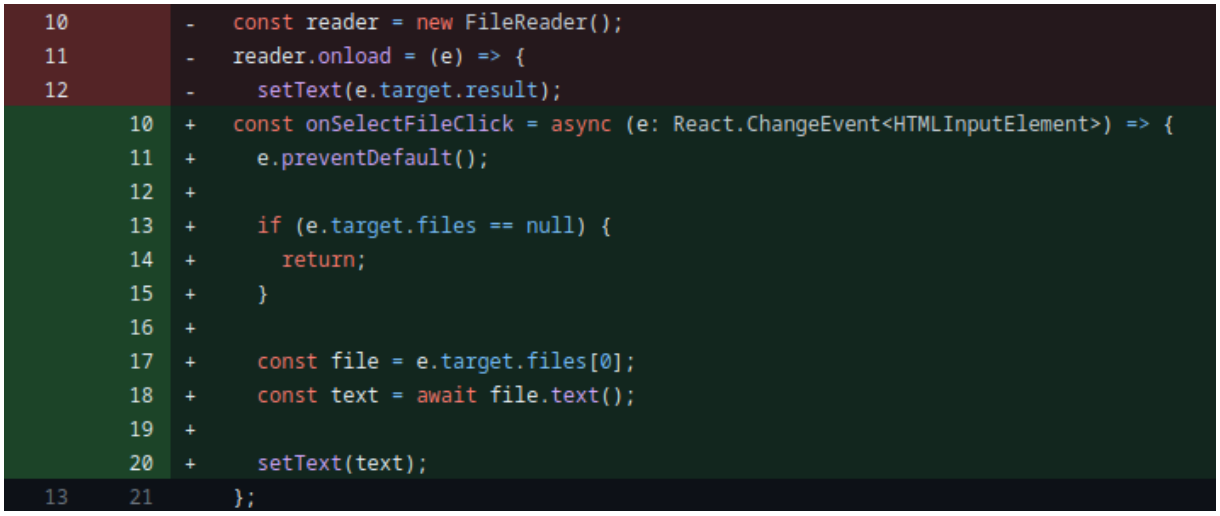
Tabela 5 – Medição dos atributos após refatorar o *smell Missing Union Type*.

Sistema	Complexidade				Tamanho		
	CC	ACC	SCC	MaxNest	LN	CLOC	CountDeclFunction
Após a refatoração do *smell Missing Union Type*	3.272	1,93	9.594	14	57.053	3.413	1.933

Em relação às métricas foram adicionadas 10 linhas de código, que correspondem a criação dos tipos, para refatorar esse *smell*. A *pull request* que corrige esse code *smell* pode ser vista aqui.

3.5 Medição 4 - Após refatorar o *smell Uncontrolled Component*

Em relação a refatoração desse *smell* foi necessário entender o que estava ocasionando esse *smell,* após entendermos do que se tratava precisamos também pesquisar sobre a File API dos navegadores. Foi refatorado 1 *smell* do tipo *Uncontrolled Component*. As imagens mostram em vermelho como estava antes da correção do *smell* e em verde como ficou após a correção do *smell*, ambas foram alterações realizadas no mesmo arquivo.



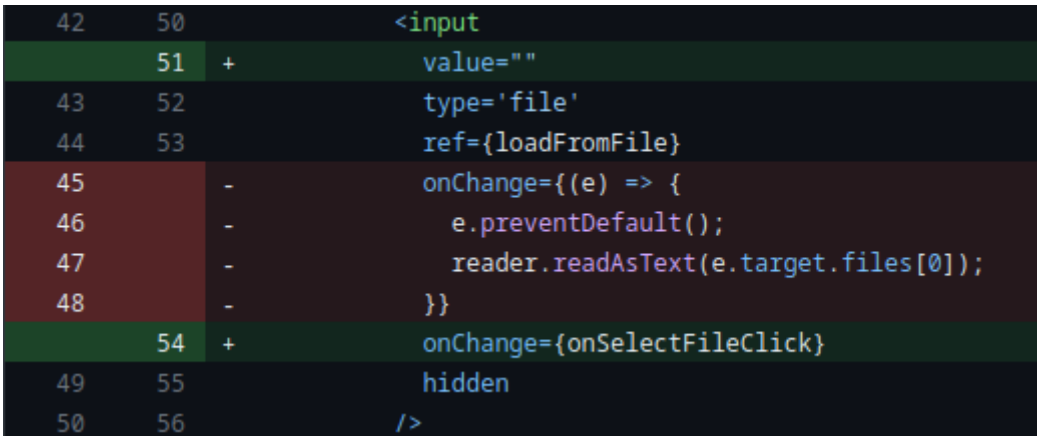


Tabela 6 – Medição dos atributos após refatorar o *smell Uncontrolled Component*.

Sistema	Complexidade				Tamanho		
	CC	ACC	SCC	MaxNest	LN	CLOC	CountDeclFunction
Após a refatoração do *smell Uncontrolled Component*	3.273	1,93	9.595	14	57.061	3.413	1.932

Durante a refatoração para corrigir esse *smell* as seguintes métricas aumentaram: CC, SCC e LN, já a métrica CountDeclFunction diminuiu. A *pull request* que corrige esse code *smell* pode ser vista aqui.

3.6 Medição 5 - Após refatorar o *smell Any Type*

Em relação a refatoração desse *smell* foi necessário apenas adicionar um valor para um input. Foram refatorados 8 *smells* do tipo *Any Type*. Ao tentar refatorar, tentamos descobrir o verdadeiro tipo para a variável. Em vermelho está o antes da alteração e em verde está como ficou após a alteração.

```
const [data, setData] = useState({ nodes: [] as any[], links: [] as any[] });
const [data, setData] = useState({ nodes: [] as Node[], links: [] as Link[] });
```

Tabela 7 – Medição dos atributos após refatorar o *smell Any Type*.

Sistema	Complexidade				Tamanho		
	CC	ACC	SCC	MaxNest	LN	CLOC	CountDeclFunction
Após a refatoração do *smell Any Type*	3.272	1,93	9.594	14	57.043	3.413	1.933

É notório que nenhuma das métricas mudou, pois a maneira como esse *smell* é refatorado não influencia nenhuma delas. A *pull request* que corrige esse code *smell* pode ser vista aqui.

4 COMPARAÇÃO DOS RESULTADOS

Conforme o artigo aponta, o atributo crítico sugerido por uma métrica de código pode variar. Isso porque, enquanto alguns atributos se tornam críticos à medida que os valores da métrica aumentam, outros se tornam críticos à medida que os valores da métrica diminuem.

Tabela 8 - Métricas para indicadores de qualidade internos e análise de criticidade. Fonte: Fernandes et al. (2020)

Attribute	Code Metric	Acronym	Granularity	Critical if
Cohesion	Lack of Cohesion of Methods [17]	LCOM2	Class	Increases
	Lack of Cohesion of Methods [21]	LCOM3	Class	Increases
	Tight Class Cohesion [19]	TCC	Class	Decreases
Complexity	Cyclomatic Complexity [20]	CC	Method	Increases
	Essential Complexity [20]	Evg	Method	Increases
	Nesting	MaxNest	Method	Increases
	Paths [22]	NPATH	Method	Increases
	Weighted Method per Class [17]	WMC	Class	Increases
Coupling	Coupling between Objects [17]	CBO	Class	Increases
	Coupling Dispersion [14]	CDISP	Method	Increases
	Coupling Intensity [14]	CINT	Method	Increases
	Fan-in [23]	FANIN	Method	Increases
	Fan-out [23]	FANOUT	Method	Increases
Inheritance	Base Classes [24]	IFANIN	Class	Decreases
	Depth of Inheritance Tree [17]	DIT	Class	Decreases
	Number Of Children [17]	NOC	Class	Decreases
	Override Ratio [14]	OR	Class	Increases
Size	Classes [15]	CDL	File	Increases
	Instance Methods [15]	NIM	Class	Increases
	Instance Variables [15]	NIV	Class	Increases
	Lines of Code [15]	LOC	Method	Increases
	Lines with Comments [15]	CLOC	Method	Decreases
	Number of Public Fields [14]	NOPA	Class	Increases
	Statements [15]	STMTC	Method	Increases
	Weight of a Class [14]	WOC	Class	Increases

Com base na tabela acima, comparamos o aumento percentual das métricas de qualidade do código do projeto Neodash antes de qualquer refatoração (que denominamos "início") com o aumento percentual das métricas nas diferentes etapas de refatoração. No final da tabela, comparamos as métricas após todas as refatorações. As comparações estão na tabela abaixo.

Tabela 8 - Comparação das métricas em comparação com o início

	Complexidade	**Tamanho**

Métricas	CC	ACC	SCC	MaxNest	LN	CLOC	CountDeclFunction
Início	3.272	1,93	9.594	14	57.043	3.413	1.933
Após a refatoração do *smell Enum Implicit Values*	3.272	1,93	9.594	14	57.043	3.413	1.933
Após a refatoração do *smell Missing Union Type*	3.272	1,93	9.594	14	57.053	3.413	1.933
Após a refatoração do *smell Uncontrolled Component*	3.273	1,93	9.595	14	57.061	3.413	1.932
Após a refatoração do *smell Any Type*	3.272	1,93	9.594	14	57.043	3.413	1.933
Métricas Finais	3.273	1,93	9.595	14	57.071	3.413	1.932

Verde indica que a métrica se manteve, e vermelho uma piora, ambos em relação às métricas iniciais. Como os code smells foram refatorados em branches independentes, a ordem das linhas não indica ordem temporal. Ao final foram coletadas as métricas novamente, juntando todas as refatorações realizadas, disponíveis na última linha.

5 REFERÊNCIAS

AZEEM, Muhammad. Machine learning techniques for code smell detection: A systematic literature review and meta-analysis. Information and Software Technology, v. 108, p. 115-138, 2019.

SABIR, Fatima. A systematic literature review on the detection of smells and their evolution in object-oriented and service-oriented systems. Software: Practice and Experience, v. 49, n. 1, p. 3-39, 2019.

Eduardo Fernandes, Alexander Chávez, Alessandro Garcia, Isabella Ferreira, Diego Cedrim, Leonardo Sousa, Willian Oizumi, Refactoring effect on internal quality attributes: What haven’t they told you yet?