



**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 | 2 |
| 2 | ISSUES DO PROJETO DE SOFTWARE LIVRE | 2 |
| 3 | AVALIAÇÃO DO PROJETO | 5 |
| 3.1 | Medição 1 – Antes de refatorar o projeto | 5 |
| 3.2 | Detecção dos Code Smells | 5 |
| 3.3 | Medição 2 - Após refatorar o smell <i>Enum Implicit Values</i> | 6 |
| 3.4 | Medição 3 - Após refatorar o smell <i>Missing Union Type</i> | 7 |

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* e mais de 650 *commits* na branch principal. Ademais, o projeto está sendo mantido atualmente e possui várias *branches* ativas, tais *branches* buscam adicionar funcionalidades para a aplicação, para mais e 31 *pull requests* esperando análise.

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.
- **Won't fix:** 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 um pouco mais complexo, então arimos o sistema e procuramos algum bug simples que conseguíssemos resolver. Após algum tempo procurando conseguimos achar um bug relacionado a *user experience*, o *bug* 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. 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

Connect to Neo4j

| | | |
|---------------------|-----------|------|
| Protocol | Hostname | Port |
| neo4j ▼ | localhost | 7687 |
| Database (optional) | | |
| neo4j | | |
| Username | | |
| neo4j | | |
| Password | | |
| ●●●●●●●● | | |

Connect ▶

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.

Add back arrow on connection modal #675

Merged nielsdejong merged 5 commits into `neo4j-labs:develop` from `hugorplobo:fix-back-arrow` last week

Conversation 2 Commits 5 Checks 5 Files changed 2

hugorplobo commented 2 weeks ago

Solves [#674](#).

fix: add back arrow on connection modal ✓ e7f0666

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 |
| 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 Ciclométrica, ACC - Complexidade Ciclométrica Média, SCC - Complexidade Ciclométrica 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 [aqui](#).

Tabela 3 – *Code smells* do projeto.

| Nome do <i>Code Smell</i> | Quantidade |
|-----------------------------|------------|
| <i>Large Component</i> | 52 |
| <i>Too many props</i> | 47 |
| <i>Any Type</i> | 12 |
| <i>Enum Implicit Values</i> | 7 |

| | |
|---|---|
| <i>Missing Union Type</i> | 5 |
| <i>Direct DOM Manipulation</i> | 2 |
| <i>Inheritance instead of composition</i> | 2 |
| JSX outside the render method | 1 |
| <i>Uncontrolled component</i> | 1 |

3.3 Medição 2 - Após refatorar o smell *Enum Implicit 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 <i>Enum Implicit Values</i> | 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.

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) => {
    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 <i>Missing Union Type</i> | 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.