

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SÃO PAULO – CAMPUS CAMPOS DO JORDÃO**

BEATRIZ HELENA E SILVA PINTO

RELATÓRIO – APLICATIVO CONSTRUÍDO EM QT E C++

CAMPOS DO JORDÃO

2025

LISTA DE FIGURAS

Figura 1 - Imagem utilizada para o background da aplicação	10
Figura 2 - Imagem de referência 1	10
Figura 3 - Imagem de referência 2	10
Figura 4 - Fonte utilizada no projeto	11
Figura 5 - Estilo Padrão	14
Figura 6 - Estilo Pomodoro.....	15
Figura 7 - Estilo Short Break	15
Figura 8 - Estilo Long Break	15
Figura 9 - Página de Configuração	15

LISTA DE SIGLAS

IFSP Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

CPP C ++ = Linguagem de programação

HPP Arquivo de interface em C++

POO Programação Orientada a Objetos

UI User Interface / Interface de Usuário

SUMÁRIO

1	INTRODUÇÃO.....	5
1.1	Objetivos.....	5
1.2	Justificativa	5
1.3	Aspectos Metodológicos	5
1.4	Aporte Teórico	5
2	O MÉTODO POMODORO.....	7
2.1	A Aplicação do Método.....	7
2.2	Customização	8
3	A CONSTRUÇÃO DA ESTRUTURA DE PASTAS.....	9
4.	OS ASSETS.....	10
4.1	Imagens	10
4.2	Fontes.....	11
5	A ESTRUTURA DO APLICATIVO	12
5.1	As Páginas.....	12
5.2	Os Botões.....	12
5.3	Os Editores de Tempo.....	13
6	PASSOS PARA CONSTRUÇÃO DA APLICAÇÃO	16
6.1	Estados do Aplicativo	16
6.2	Fluxo e controle	17
6.3	Idioma	18
7	DESAFIOS E OBSERVAÇÕES.....	19
8	CONCLUSÃO.....	21
	BIBLIOGRAFIA	22

1 INTRODUÇÃO

De acordo com Gillis (2024), a programação orientada a objetos é um paradigma centrado nos objetos, sendo que esse possui características e comportamentos próprios. BasuMallick (2022), explana as aplicações na qual a programação orientada a objetos pode ser usada, dentre elas, desenvolvimento web, desenvolvimento de jogos, múltiplos softwares com diferentes funcionalidades, dentre outros. Ainda, Herrity (2025) destaca as estruturas desse paradigma de programação: Classes; Atributos; Métodos; e Objetos. As classes funcionam como “*blueprints*” dos objetos, os modelos que demonstram como é aquela classe, mas não especifica mais (Gillis, 2024).

1.1 Objetivos

Desenvolver um aplicativo usando o QT Creator em conjunto com a programação orientada a objetos em C++.

1.2 Justificativa

Para ampliar os conhecimentos referentes a programação orientada a objetos, uma das formas eficazes é construir aplicações simples de forma a sintetizar os conhecimentos e traduzi-los em uma aplicação funcional, aplicando melhorias consecutivamente e consertando possíveis erros, dessa forma, desenvolver uma aplicação usando o QT além de apresentar uma nova ferramenta, bastante visual, treina os conhecimentos adquiridos em sala.

1.3 Aspectos Metodológicos

Para concepção desse trabalho, além da documentação feita por meio deste relatório, foram elaborados um pequeno documento referente ao próprio software, utilizando a ferramenta Notion, o objetivo principal foi listar as funcionalidades, verificar as páginas que seriam criadas e armazenar os detalhes do design.

1.4 Aporte Teórico

Como aporte teórico para elaboração desse projeto, consideram-se:

- As aulas teóricas;
- DEITEL, Harvey M.; DEITEL, Paul J. C++: como programar. Tradução de Edson Furmankiewicz. Revisão técnica de Fábio Luis Picelli Lucchini. 3. reimpr.

São Paulo: Pearson Education do Brasil, 2010. (Disponível na biblioteca do Campus)

- O conteúdo disponibilizado no site oficial da QT: a documentação e curso.

2 O MÉTODO POMODORO

Francesco Cirilo, o italiano criador do método, desenvolveu a técnica nos anos 1980, em seu livro *The Pomodoro Technique* (2007), o autor explica que dentre os objetivos da técnica, destacam-se: o aumento do foco, da motivação e constância, melhora o processo de trabalho/estudo, dentre outras vantagens. Sheldon e Wigmore (2022), explicam que o italiano se baseou em sua organização pessoal durante seus estudos na faculdade para criar a técnica e melhorá-la.

Cirilo (2007) explica que o método consiste em 5 etapas: o planejamento, a aplicação do método, o registro, o processamento e a visualização. O planejamento das atividades acontece no início do dia, o registro, processamento e visualização são feitos no final do dia e composto por uma compilação de observações, o processamento para transformar os dados em informação e um ajuste das informações de forma clara para definir melhorias e caminhos.

Para que a técnica funcione é importante entender que é preciso construir uma relação com o tempo, de maneira incremental, que envolve desde estabelecer uma rotina diária e, posteriormente, semanal, concentrar-se para alcançar os objetivos e adaptar a técnica conforme necessário e para equipes. (Sheldon, Wigmore, 2007)

2.1 A Aplicação do Método

Gupta (2025) define a seguinte sequência de ações para a prática da técnica:

- Selecionar uma tarefa para o pomodoro;
- Fixar um relógio com 25 minutos;
- Trabalhar na tarefa durante esse tempo;
- Fazer uma pausa (breve) de 5 minutos;
- Repetir esse formato 3 vezes.

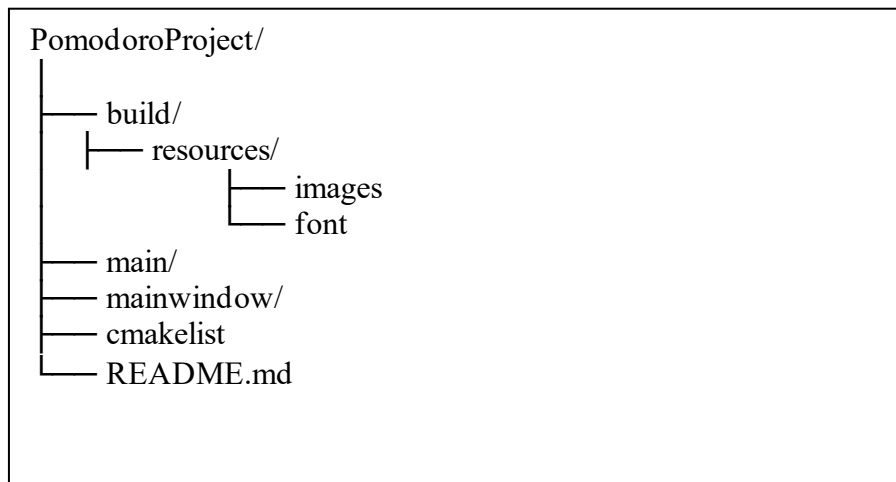
- Após o 4º tempo de pomodoro, fazer uma pausa (longa) de 15 a 30 minutos.

2.2 Customização

O modelo padrão, desenvolvido por Francesco Cirilo, propõem o estilo, 25-5-15, no entanto essa padronização do tempo, pode não ser adequado a todos os estilos de trabalho/estudo, dessa forma o método pode passar por uma adaptação para ser utilizado, sessões mais longas de 50 minutos com pausas de 10 minutos, é uma das opções para indivíduos que desejam ou conseguem manter-se focados por mais tempo em uma tarefa (Anderson, 2024).

3 A CONSTRUÇÃO DA ESTRUTURA DE PASTAS

Para construção da aplicação, usou-se o modelo básico de projeto disponível no QT do tipo “*QT Widget Application*”, o qual gera uma estrutura de pastas básica e os documentos básicos necessários para a criação da aplicação, tudo isso com os caminhos prévios de pastas. Acrescentaram-se, também, as pastas de *Resources* (recursos), *Images* (Imagens) e *Fonts* (Fontes), para armazenar sons, imagens, fontes e quaisquer outros recursos de estilo necessários para o desenvolvimento do projeto.



4. OS ASSETS

4.1 Imagens

A única imagem externa utilizada foi a do *background*, a qual foi gerada com auxílio de uma IA enviando duas referências que serviram de base para criação.



Figura 1 - Imagem utilizada para o background da aplicação



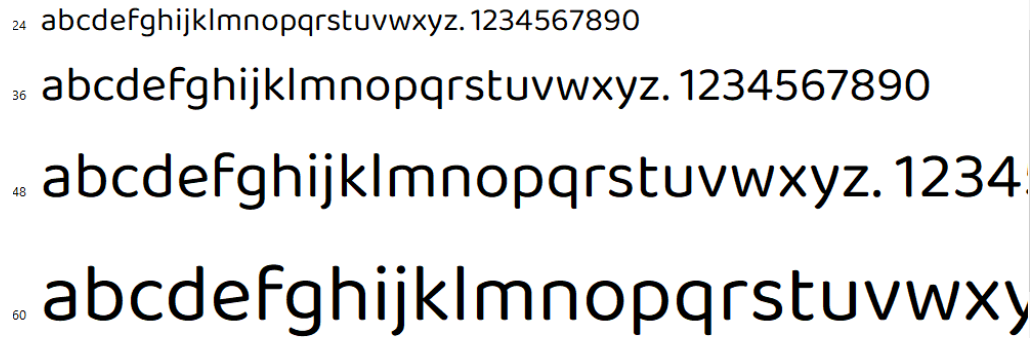
Figura 2 - Imagem de referência 1



Figura 3 - Imagem de referência 2

4.2 Fontes

A fonte escolhida, Baloo2, é uma fonte pronta, com direitos livres, leve e com bem desenhada para combinar com a estética do projeto.



24 abcdefghijklmnopqrstuvwxyz. 1234567890

36 abcdefghijklmnopqrstuvwxyz. 1234567890

48 abcdefghijklmnopqrstuvwxyz. 1234

60 abcdefghijklmnopqrstuvwxyz

Figura 4 - Fonte utilizada no projeto

5 A ESTRUTURA DO APLICATIVO

Para a concepção da aplicação considerou-se o foco direto no Timer da técnica, um controle automático, em loop para o usuário, mas com a funcionalidade de pausa, recomeçar e a personalização de acordo com as necessidades do usuário.

5.1 As Páginas

O primeiro passo foi montar as duas páginas, da forma como foi previsto no planejamento, no primeiro momento a tentativa foi feita usando o *TabWidget* do QT, mas a tela ficou muito poluída com a construção dessa construção, a barra padrão de aplicativos, a de configurações e as duas abas, embora ficasse funcional, ficou visualmente desagradável, então foi feita a troca para o *StackedWidget*, a alteração na parte referente ao código foi simples, visto que o *StackedWidget* funciona como uma pilha de *array* em suas páginas (a primeira corresponde ao index 0 e as demais seguem a sequência numérica) como no código abaixo, no qual mostra a configuração para o mesmo botão – o qual volta para a página principal, Pomodoro - com os dois métodos .

```
1 //Usando o TabWidget
2 void MainWindow::on_btnGoBack_clicked() {
3     ui->tabWidget->setCurrentWidget(ui->pagePomodoro);
4 }
5 //Usando o StackedWidget
6 void MainWindow::on_btnGoBack_clicked() {
7     ui->Pomodoro->setCurrentIndex(0);
8 }
9
```

5.2 Os Botões

O QT oferece uma variedade de botões pré-prontos, o que dá certa facilidade para o desenvolvimento em si, basta posicioná-los no local onde deseja que eles fiquem no UI (*User Interface* – Interface do Usuário), e adicionar o que eles fazem por

meio dos comandos: botão direito -> go to slots. O Qt cria o método no *mainwindow* (janela principal) onde podemos implementar os controles, e também, adiciona automaticamente esse método ao arquivo de interface.

5.3 Os Editores de Tempo

São, praticamente, o *core* da aplicação, o QT já possui um modelo pré-pronto para a visualização do usuário, o *QTimeEdit*, para o projeto foram considerados 3 editores, o Pomodoro referente ao tempo desejado para foco, o *Short Break* (pausa curta) para as pausas menores e o *Long Break* (pausa longa) para os momentos maiores de descanso.

Para implementação, no entanto, foram necessários alguns ajustes:

- Edição de horas:minutos para minutos:segundos : Durante os primeiros testes, notou-se que os timers estavam configurados previamente para um modelo que representa horas e minutos, no entanto, como a técnica do Pomodoro, a princípio, utiliza o tempo em minutos, foi necessário alterar a configuração inicial do item *TimeEdit*, o qual estava configurado no padrão horas: minutos (hh:mm), alterando para minutos:segundos (mm:ss).

```
1 // Configura todos os QTimeEdit para mostrar minutos:segundos
2     ui->timePom->setDisplayFormat("mm:ss");
3     ui->timeSB->setDisplayFormat("mm:ss");
4     ui->timeLB->setDisplayFormat("mm:ss");
```

- Tempo padrão: Um detalhe importante é que esse aplicativo visa ser customizável, ou seja, o usuário pode alterar o tempo, mas a princípio, a aplicação possui um tempo padrão, o qual é o mesmo indicado na técnica, sendo um Pomodoro de 25 minutos, pausas curtas de 5 minutos e a cada 4 pomodoros, uma pausa longa de 15 minutos.

```

1 // Define valores padrão
2 ui->timePom->setTime(QTime(0, 25, 0)); // 25 minutos
3 ui->timeSB->setTime(QTime(0, 5, 0)); // 05 minutos
4 ui->timeLB->setTime(QTime(0, 15, 0)); // 15 minutos

```

- Atualização da Interface: Como dito, a aplicação visa ser customizável, dessa forma, quando um usuário altera os valores padrões de tempo, o aplicativo deve responder visualmente, por meio da interface, assim, a terceira etapa de configuração dos *QTimeEdit* é a atualização da UI para mostrar os novos tempos, que foi feita dentro do método *btnGoBack* (botão de voltar):

```

1 // Atualiza o display com o tempo configurado
2 QTime displayTime;
3 switch(currentMode) {
4 case POMODORO: displayTime = ui->timePom->time(); break;
5 case SHORT_BREAK: displayTime = ui->timeSB->time(); break;
6 case LONG_BREAK: displayTime = ui->timeLB->time(); break;
7 }

```

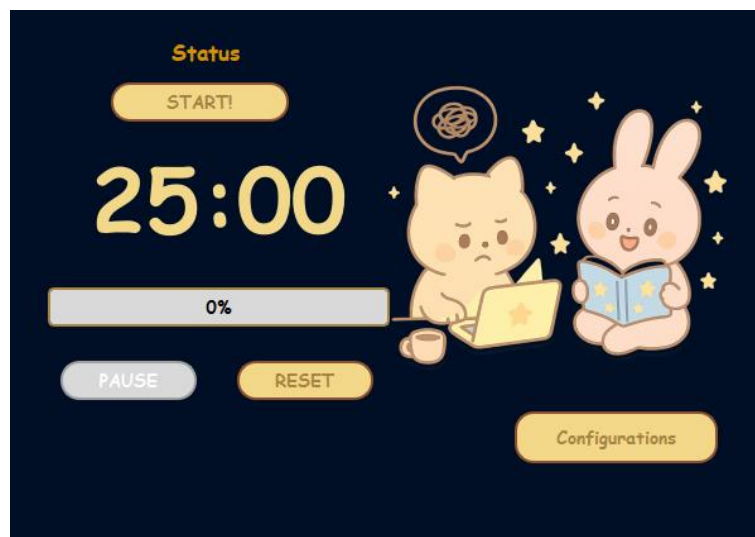


Figura 5 - Estilo Padrão

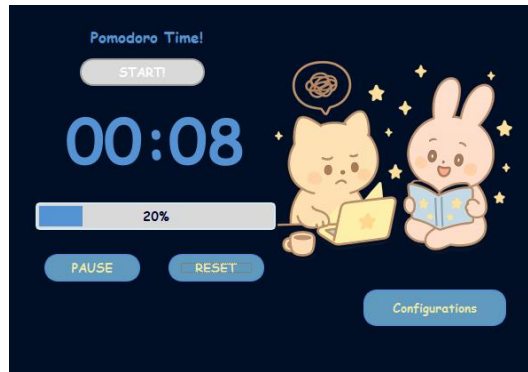


Figura 6 - Estilo Pomodoro

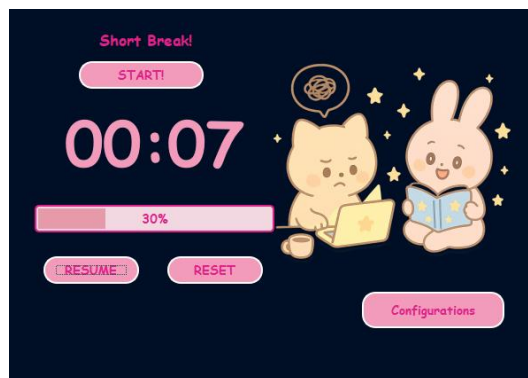


Figura 7 - Estilo Short Break

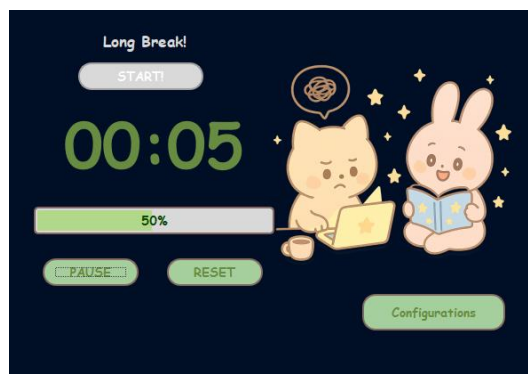


Figura 8 - Estilo Long Break

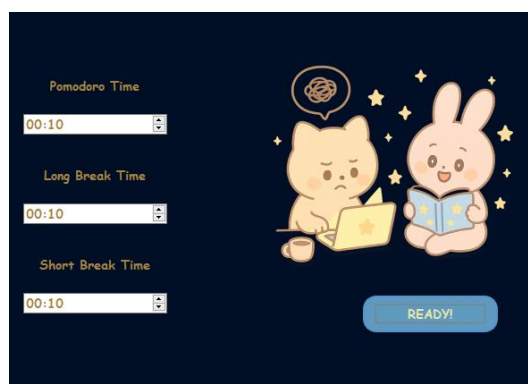


Figura 9 - Página de Configuração

6 PASSOS PARA CONSTRUÇÃO DA APLICAÇÃO

Para realizar a construção desse aplicativo, considerou-se primeiro, as funções principais de um relógio Pomodoro: O relógio de tempo, o qual deveria ser customizável para que o usuário altere conforme suas necessidades, o mesmo para os relógios de pausa: longa e curta (*Long Break* e *Short Break*), além desses os botões de iniciar, pausa e recomeçar (*Start*, *Pause* e *Reset*). A partir disso, visou-se montar o aplicativo por camadas, com essas sendo a base, de forma a facilitar incrementos e outras funcionalidades adiante como som, barra de progresso visual e etc.

6.1 Estados do Aplicativo

Como em outro projeto, usar “Estados” para controlar o que acontece na aplicação se mostrou um jeito claro e simples de realizar controlar a interface e os processos, o mesmo, foi aplicado para esse, com a variável “*Current_Mode*” (modo atual) a qual armazena 3 estados:

- **Pomodoro:** Modo padrão e que fica ativo enquanto o usuário está na tela principal sem iniciar a aplicação.
- **Short_Break:** A pausa curta, ocorre quando um ciclo do pomodoro é completo, se inicia na sequência automaticamente.
- **Long_Break:** Também automática, mas se inicia quando 4 ciclos de pomodoro (3 pausas curtas) são completos.

```
1 //Switch que troca o tempo conforme o estado de jogo
2 switch(currentMode) {
3     case POMODORO: displayTime = ui->timePom->time(); break;
4     case SHORT_BREAK: displayTime = ui->timeSB->time(); break;
5     case LONG_BREAK: displayTime = ui->timeLB->time(); break;
6 }
```


6.2 Fluxo e controle

O fluxo entre os estados é controlado por tempo, visto que os estados de pausa, sejam elas longas ou curtas, podem apenas ser acessadas quando o ciclo do estado Pomodoro é finalizado. Esse controle foi feito através da função *startNextSession* (iniciar nova sessão) com uma série de laços e um contador do estado pomodoro (*pomodoroCount*), o qual só é incrementado quando o *pomEnd == True*, ou seja, quando é constatado que o timer do pomodoro chegou a zero.

```

1 //Inicia nova sessão de pomodoro
2 void MainWindow::startNextSession()
3 {
4     //para o timer
5     pomodoroTimer->stop();
6
7     // Só muda de estado se o pomEnd == true
8     /*Para entender SB e LB só acontecem quando se completa um ciclo
9 de pomodoro*/
10    if (pomEnd) {
11        if (currentMode == POMODORO) {
12            pomodoroCount++;
13            //para trocar entre SB e LB - 4 pomodoros = 1 LB
14            //checa se pomodoroCount é igual a 4 por divisão
15            if (pomodoroCount % POM_FOR_LB == 0) {
16                currentMode = LONG_BREAK;
17                ui->labelStatus->setText("Long Break Time!");
18            } else {
19                currentMode = SHORT_BREAK;
20                ui->labelStatus->setText("Short Break Time!");
21            }
22        } else {
23            currentMode = POMODORO;
24            ui->labelStatus->setText("Pomodoro Time!");
25        }
26        pomEnd = false;
27    }

```

A variável `POM_FOR_LB` é uma constante definida especificamente para esse cálculo de checagem para a *Long Break*.

6.3 Idioma

Por ser um aplicativo simples, a opção inicial foi fazê-lo em inglês, considerando que pode atingir mais usuários, também, para executar futuros testes dos arquivos de tradução do próprio QT.

7 DESAFIOS E OBSERVAÇÕES

Toda ferramenta nova tem potencial para apresentar desafios, surpreendentemente, o QT Creator, tem surpresas positivas além dos desafios de seus comandos e sua interface. Um detalhe sutil que, no entanto, auxilia demais no desenvolvimento dentro da aplicação é a criação da UI visual para o desenvolvedor, mover os botões, ver como a aplicação vai ficar desde o princípio e poder testar isso, ajuda a compreender melhor a interação de usuário x aplicação e enxergar as melhorias que talvez passassem despercebidas numa aplicação a qual não se tem o visual. Como a aplicação foi montada em pequenas camadas visando a sua funcionalidade, durante o percurso de criação, pode-se notar as várias necessidades de acréscimos para a melhoria do aplicativo: detalhes não considerados inicialmente, como a necessidade de edição dos timers após configuração, a barra de progresso e o tamanho dos itens em tela, foram facilitados de forma extrema, bem como, a melhoria e teste do design do aplicativo, o que permitiu que fossem criados mais de um arquivo de estilo para os diferentes estados do aplicativo.

Em contrapartida, o desconhecimento dos comandos específicos e de como as coisas dentro da aplicação funcionam, a princípio, se demonstram um desafio para iniciar a programação, detalhe que se torna mais leve ao longo do projeto, visto que ao final do desenvolvimento, parece já estar gravado. A dificuldade em aplicar algumas mudanças também foi um problema, em especial com relação aos estilos, onde colocar a pasta de recursos, os caminhos relativos até eles, as limitações do QSS (QT Style Sheet) quanto aos recursos disponíveis, tornaram mais complexa a implementação.

Outro detalhe, é que embora a lógica possa ser exercitada durante o processo de criação, é preciso ficar atento a dinâmica do QT, ao usar a página UI que vem na

aplicação, o próprio QT registrar os métodos na classe, e os cria no arquivo mainwindow, restando ao desenvolvedor a lógica de cada método, o que para quem não está acostumado com POO ou mesmo com o aplicativo, se torna um pouco confuso.

A integração com recursos externos e a necessidade de instalar mais recursos internos (dependências), embora complexa a princípio, oferece uma visualização mais completa do software e das diversas funcionalidades, o que é mais um ponto positivo, principalmente considerando que é uma nova ferramenta.

8 CONCLUSÃO

A ferramenta QT é muito versátil e interessante e além de positiva para os estudos, ela se mostra simples de aprender e possibilita uma série de aplicações depois que se conhece sua interface e seu funcionamento. Com relação a programação orientada a objetos, ela funciona como uma guia prática, é preciso compreender a lógica por trás do paradigma e seus pilares para que a aplicação se mostre como ferramenta de aprendizado e não fique confusa durante o desenvolvimento.

No quesito de funcionalidades futuras, o software oferece uma série de funcionalidades interessantes, para esse aplicativo, o aumento da tela, um implementador de tarefas e a integração com uma rádio são alternativas futura interessantes para tornar a aplicação mais completa e versátil.

Quanto ao aprendizado durante o desenvolvimento da aplicação e a linguagem utilizada, o conhecimento prévio de C++ se mostra como uma vantagem, facilitando o desenvolvimento, visto que já se conhece a sintaxe e tudo fica mais claro, mesmo quando essa se une aos comandos próprios do QT Creator.

Enfim, a ferramenta é interessante para criações futuras, simples e também mais complexas, embora apresente desafios na busca de pacotes internos e dependências do próprio QT, ainda assim, é uma forma visual e diferente de desenvolver.

BIBLIOGRAFIA

ANDERSON, C. **What is pomodoro technique adaptations? – Focuskeeper Glossary**. Disponível em: <<https://focuskeeper.co/glossary/what-is-pomodoro-technique-adaptations>>. Acesso em: 20 maio 2025.

CIRILLO, F.; INFO, B. **The Pomodoro Technique (The Pomodoro)**. [s.l.: s.n.]. Disponível em: <<https://www.northbaycounselling.com/wp-content/uploads/2022/05/Cirillo-Pomodoro-Technique.pdf>>. Acesso em: 25 maio 2025.

DEITEL, Harvey M.; DEITEL, Paul J. **C++: como programar**. Tradução de Edson Furmankiewicz. Revisão técnica de Fábio Luis Picelli Lucchini. 3. reimpr. São Paulo: Pearson Education do Brasil, 2010. (Disponível na biblioteca do Campus)

GUPTA, S. **What Is the Pomodoro Technique?** Disponível em: <<https://www.verywellmind.com/pomodoro-technique-history-steps-benefits-and-drawbacks-6892111>>. Acesso em: 25 maio 2025.

SHELDON, R.; WIGMORE, I. **What Is Pomodoro technique?** Disponível em: <<https://www.techtarget.com/whatis/definition/pomodoro-technique>>. Acesso em: 21 maio 2025