

# Kata Kid - Ensinando a como criar uma Extensão para o Visual Studio Code

Eliedson de Souza<sup>1</sup>

Carlos Mario Dal Col Zeve

<sup>1</sup>Instituto Federal de Educação Ciência e Tecnologia do Rio Grande do Sul (IFRS) Rodovia RS 135, Km 32,5 — Distrito Eng. Luiz Englert — CEP: 99170-000 — Sertão/RS

0101222@aluno.sertao.ifrs.edu.br

**Abstract.** *This paper aims to present and detail the project carried out for the Final Course Work, which describes how to create an extension for Visual Studio Code, in addition to showing the development done for the extension, where the user can log in to be able to record the coding activities carried out, allowing the same to monitor their progress in different programming languages, save the time that was invested, make use of the Pomodoro study technique and see the daily, weekly and monthly statistics of their progress as a developer, in addition to creating their own goals. The article also includes the research that was done to the requirement discovery that were used and diagrams developed for better understanding of it. Finally, in the conclusion, the results and suggestions of this project will be mentioned, which becomes useful to the academic environment and society.*

**Resumo.** *O presente trabalho tem a finalidade de apresentar e detalhar o projeto realizado para o Trabalho de Conclusão de Curso, onde é descrito como criar uma extensão para o Visual Studio Code, além de mostrar o desenvolvimento feito para a extensão, onde o usuário pode fazer login para que possa registrar as atividades de codificação feitas, permitindo ao mesmo monitorar seu progresso em diferentes linguagens de programação, salvar a marcação de tempo que foi investido, fazer uso do modo de estudo Pomodoro e ver as estatísticas diária, semanal e mensal do seu progresso como desenvolvedor, além de criar suas próprias metas. O artigo também conta com a pesquisa que foi feita para o levantamento dos requisitos que foram usados e os diagramas elaborados para melhor entendimento do mesmo. Por fim, na conclusão serão mencionados os resultados e sugestões desse projeto a qual torna-se útil ao meio acadêmico e sociedade.*

## 1. Introdução

A tecnologia no mercado de trabalho vem crescendo nos últimos anos, como mostra o site de notícias Insper (2023): “Só em 2022, o setor de TI cresceu 22,9% no Brasil comparado a 2021, que já vinha de um crescimento de 23% sobre 2020.”. Além disso, uma pesquisa trimestral feita por Dagoberto Hajjar (2024), mostrou que o mercado de

TI fechou o primeiro trimestre de 2023 com 20% de crescimento, bem acima das expectativas para o período, já que o esperado era ser um trimestre fraco. Muitos empregos acabam por se utilizar da tecnologia, e a programação é uma das áreas incluídas dentro deste campo.

É muito importante ficar por dentro das tecnologias usadas na área [CNN, 2023]. Uma muito utilizada é o Git e também o GitHub, as quais são duas ferramentas essenciais para o desenvolvimento de software. O Git é um sistema de controle de versão distribuído, enquanto o GitHub é uma plataforma de hospedagem de repositórios Git [ARAÚJO, 2023]. Os iniciantes na área de programação e desenvolvimento de códigos podem ter certas dificuldades para aprender a lidar com a tecnologia, ainda mais pela tecnologia ser descentralizada e complicada de se usar [BAUERMANN 2020], por causa disso torna-se difícil o registro de códigos e o desenvolvimento dos programadores. Outra tecnologia que é usada para programar é o Visual Studio Code, que é um editor de código-fonte que infelizmente não possui atualizações o suficiente para monitorar o progresso de usuários, como por exemplo, um cronômetro para marcar a quantidade de tempo que o programador passa criando seu código. Tal ferramenta poderia ser útil na questão de facilitar o monitoramento de código além de cronometrar o tempo que o programador passou fazendo o código, entre outros.

Por isso, pensando nos programadores, este projeto está sendo desenvolvido com o intuito de ajudar os mesmos e ensinar como fazer uma extensão personalizada no Visual Studio Code, para que os novos programadores possam auxiliar a comunidade que utiliza a ferramenta a deixá-la com mais recursos para ajudarem outras pessoas nesse mesmo caminho. O presente trabalho vai mostrar como criar uma extensão, além de mostrar os benefícios que esta extensão pode trazer, como registrar o desenvolvimento de seus trabalhos e esforços de maneira mais eficaz. A ideia é criar uma extensão para o Visual Studio Code, que registre suas atividades de codificação. Para isso, o usuário precisa se registrar na extensão que o habilita a salvar seus dados, por isso a plataforma de Backend-as-a-Service (BaaS) conhecida como Firebase, que fornece infraestrutura de back-end pronta para quem desenvolve aplicativos [RIBEIRO 2023], foi usado, mais especificamente o Firebase Authentication o qual fornece bibliotecas de IU (Interface do Usuário) para autenticar usuários para a extensão [FIREBASE 2014b], e também o Firebase Realtime Database que é um banco de dados na nuvem NoSQL capaz de armazenar e sincronizar dados [FIREBASE 2014a]. Tais ferramentas foram usadas para fazer a autenticação dos usuários e permitir que os usuários da extensão possam salvar esses dados e assim monitorar seu progresso de desenvolvimento como programador com a extensão. A extensão possui um comando que torna possível a utilização do Método Pomodoro, criado por Cirillo (2013), o qual é um método de estudos muito usado para ajudar o usuário que a estiver utilizando para ter concentração e foco no que está fazendo. A ideia de implementar o método dá ênfase para a extensão já que ela possui o intuito de ajudar.

O presente artigo conta com 6 seções que apresentam as etapas do desenvolvimento do trabalho. Na seção 2 será explicado sobre o problema que a extensão busca auxiliar e a justificativa para a criação da mesma para programadores. A seção 3 expõe os objetivos propostos para a criação da extensão, bem como o objetivo

geral e os objetivos específicos da mesma. Na seção 4 são mencionados os trabalhos usados para embasar a pesquisa. Na seção 5 será mostrada a metodologia de como criar uma extensão e como a mesma foi criada. A seção 6 apresenta considerações finais do projeto.

## **2. Definição do Problema e Justificativa**

Quando o assunto de criar uma extensão para um aplicativo ou ferramenta é abordado, normalmente parece ser algo muito complicado de se lidar no começo, porém com um pouco de pesquisa as coisas se tornam mais fáceis. Por isso, a ideia de criar essa extensão é justamente fruto desse medo por achar ser algo complicado. A criação dessa extensão está documentada para que seja um guia para programadores, que sentem vontade de criar recursos adicionais para uma ferramenta, mas não sabem como iniciar.

Primeiro a extensão foi baseada em um dos principais problemas que os desenvolvedores enfrentam, que é a questão de registrar e monitorar o seu progresso. Muita gente aprende a usar Git e GitHub para fazer esse registro, mas mesmo assim é algo que para iniciantes se torna um pouco complicado, por isso, a ideia é criar uma maneira de registrar o seu desenvolvimento de maneira fácil e simples, podendo ver também as estatísticas e evolução que o programador teve. Outra questão é de poder cronometrar o tempo que o programador passa para criar seus códigos e sistemas, além de ajudar o usuário com um método de estudo e fazendo com que o mesmo possa definir metas para serem cumpridas enquanto programa.

Atualmente, o Visual Studio Code não possui esses recursos para que isso seja facilitado, mas pode-se criar extensões que ajudam o desenvolvedor nesses problemas. A extensão que foi feita ajuda o usuário a registrar suas atividades enquanto programa, podendo saber quanto tempo passou codificando, seja com cronômetro ou com o modo de Estudo Pomodoro, ou até mesmo o número de linhas de código feitas com determinada linguagem de programação (seja ela Python, Javascript, entre outras).

## **3. Objetivo Geral**

Desenvolver uma extensão para o Visual Studio Code na qual será usada para que o usuário registre atividades de codificação, permitindo que o usuário monitore seu progresso em diferentes linguagens de programação e o tempo usado.

### **3.1 Objetivos Específicos**

Aqui encontram-se os objetivos específicos do projeto:

- Detalhar como criar uma extensão para a ferramenta Visual Studio Code;
- Registro de Atividades:
  - Contar linhas de código escritas por linguagem de programação (Javascript, Python, etc.);
  - Rastrear o tempo gasto em cada sessão de codificação.
- Feedback e Metas:

- Permitir que o usuário defina metas de codificação (ex: escrever 500 linhas por semana);
  - Enviar lembretes ou sugestões com base no desempenho.
- Perfil do Usuário:
  - Criar um painel visual onde o usuário pode visualizar suas estatísticas de codificação (linhas escritas, tempo, etc.);
  - Comparar desempenho ao longo do tempo (diário, semanal, mensal). Além de mostrar a linguagem de programação mais usada.
- Modo de Estudo:
  - Criação de uma função que é um modo de estudo onde o usuário pode se concentrar e acompanhar o progresso.

## 4. Trabalhos Relacionados

Para embasar a pesquisa, foi revisada a literatura existente sobre a criação de extensões para o Visual Studio Code e outros trabalhos que juntam técnicas de estudos eficazes para a aprendizagem, nisso foram identificados os seguintes trabalhos:

A extensão Kata Kid é inspirada na extensão WakaTime (2025) que rastreia automaticamente a produtividade de desenvolvedores, registrando tempo de codificação, arquivos acessados e linguagens utilizadas. Na Imagem 1 e 2 podemos ver as estatísticas da extensão na web.

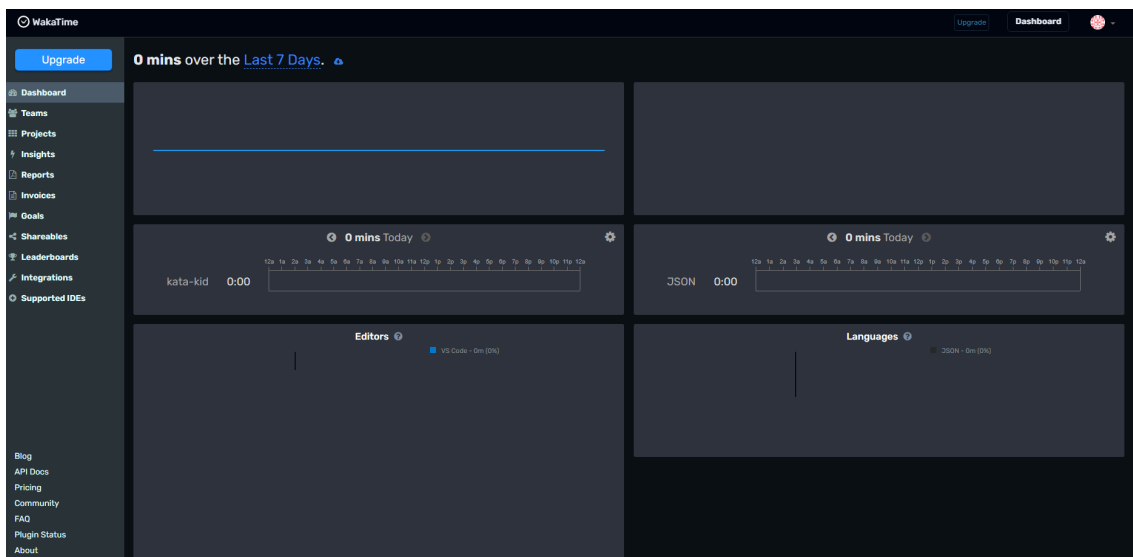
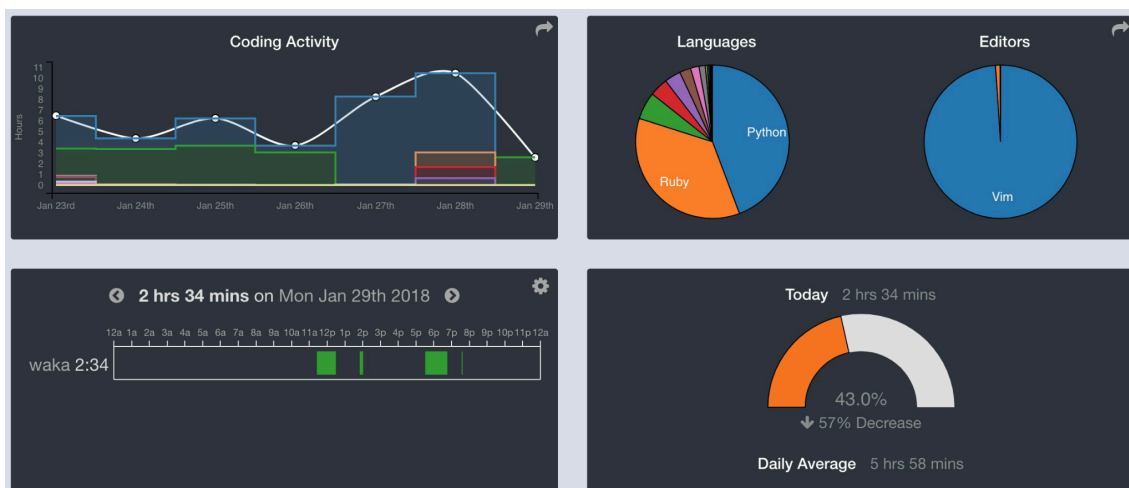


Imagem 1. Estatísticas da Extensão WakaTime na web.



**Imagem 2. Estatísticas da Extensão WakaTime.**

O artigo de Barbara Dyna Vendrameto (2022) conta sobre o Desenvolvimento de um site e uma extensão em TypeScript para uso no Visual Studio Code, com dois programas adicionais em JavaScript, como apoio para leitura e complementação de códigos.

O artigo de Andrade (2020), aborda a criação de extensões para o Visual Studio Code. Ele discute o que são extensões, quais motivos levam alguém a criar uma e as etapas básicas envolvidas. Alguns dos pontos importantes são que existem duas linguagens de programação principais para extensões, Javascript e Typescript, e que há uma API fornecida pelo Visual Studio Code para auxiliar os desenvolvedores.

O artigo "Tempo Universitário: Foco na Jornada Acadêmica" de Saraiva et al. (2024) apresenta os resultados de uma oficina desenvolvida na Universidade de Brasília (UnB) com o objetivo de aprimorar a gestão do tempo e o desempenho acadêmico dos estudantes universitários. A oficina foi estruturada em três etapas principais: definição de prioridades, manutenção do foco e aplicação de técnicas de estudo, todas embasadas por métodos consolidados na literatura, como a Matriz de Eisenhower e o Método Pomodoro.

Entre os resultados, destacou-se a eficácia do Método Pomodoro em aumentar a concentração e a produtividade dos participantes, além de práticas como mindfulness para redução da ansiedade. Os estudantes relataram melhorias significativas em sua capacidade de priorizar tarefas, organizar o tempo e aplicar métodos estruturados de estudo, resultando em aumento do desempenho acadêmico e redução da procrastinação. A pesquisa sugere que programas semelhantes podem ser replicados em outros contextos para promover maior autonomia na gestão do tempo e preparação para desafios futuros.

O trabalho de Oliveira (2024), apresenta o desenvolvimento de uma extensão para o Visual Studio Code, com o objetivo de aumentar a produtividade no processo de desenvolvimento de software. A proposta aborda os desafios enfrentados por desenvolvedores, como a necessidade de minimizar erros, otimizar a escrita de código e

aprimorar a organização e busca por informações dentro do ambiente de desenvolvimento.

A extensão desenvolvida incorpora funcionalidades voltadas para automação de tarefas repetitivas, sugestões de código baseadas em boas práticas e melhoria na navegação entre diferentes trechos do código. Além disso, ela integra um sistema de recomendações para resolver problemas recorrentes e reduzir o tempo gasto em pesquisas externas.

Os resultados apresentados indicam que o uso da extensão pode impactar positivamente a eficiência dos desenvolvedores, especialmente em projetos que exigem alta produtividade e manutenção contínua. O trabalho também discute aspectos de design da interface e a experiência do usuário, enfatizando a importância de ferramentas intuitivas e bem integradas ao fluxo de trabalho.

O trabalho de Oliveira (2024) é relevante como referência para pesquisas futuras que buscam explorar soluções tecnológicas para aumentar a produtividade de equipes de desenvolvimento, bem como para profissionais interessados em incorporar boas práticas e automação ao dia a dia de trabalho. Além disso, sua proposta se relaciona diretamente com o presente artigo, que também aborda o desenvolvimento de uma extensão para o Visual Studio Code. Neste caso, o foco é auxiliar programadores a criarem suas próprias extensões do zero, com funcionalidades úteis tanto para o aprendizado quanto para acompanhar o progresso do desenvolvimento, destacando o potencial da ferramenta para fins educacionais e profissionais.

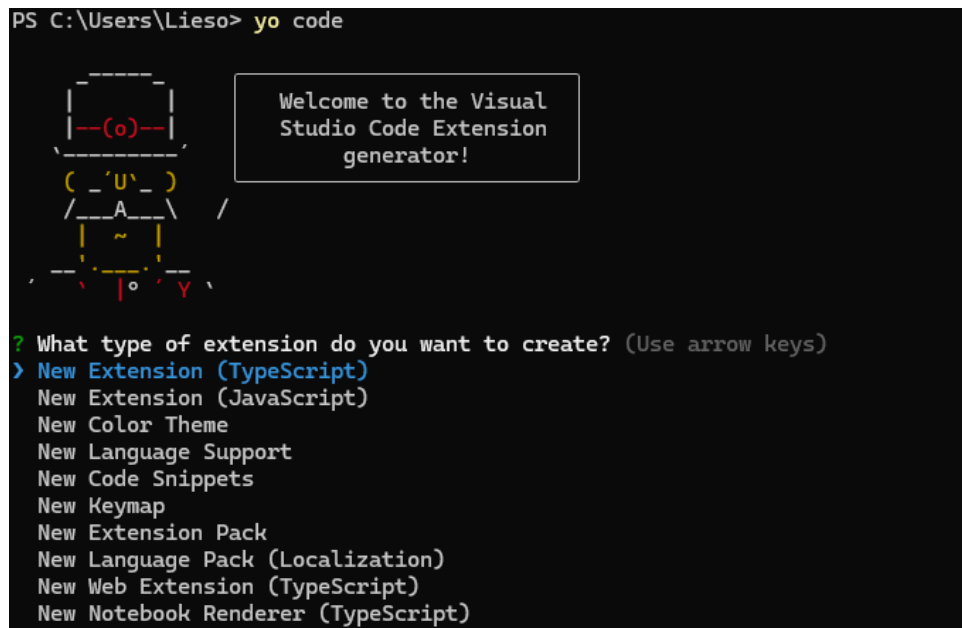
Além desses artigos e extensões, o livro *The Pomodoro Technique*, criado por Cirillo (2013), foi utilizado como embasamento por causa que a extensão possui o gerenciamento de tempo, que é um princípio do método. Mesmo a técnica tendo sido criada no final dos anos 1980, é um método de gerenciamento de tempo que visa aumentar a produtividade e o foco. Ela se baseia na divisão do tempo de trabalho em blocos de 25 minutos, chamados de "pomodoros", intercalados por pausas curtas.

## 5. Metodologia

### 5.1 Instalação

De acordo com Andrade (2020), torna-se importante começar a criar uma extensão para o Visual Studio Code [Microsoft 2024a], instalando o mesmo em seu computador, além do Node.js que permite a execução de códigos Javascript fora de um navegador web [Ryan Dahl 2009]. Os links para download podem ser encontrados nas referências. No terminal do Windows, pode ser usado o comando: `npm install -g yo generator-code`, para que faça a instalação do gerador de extensões. O comando usado instala globalmente duas ferramentas, o Yeoman que ajuda a criar rapidamente a estrutura inicial de diferentes tipos de projeto e o Generator-code que é um gerador específico para criar extensões do Visual Studio Code (Imagem 3). O `-g` no comando `npm install` significa que o pacote será instalado globalmente no seu sistema, e

não apenas no diretório atual do projeto. Em seguida com o comando: `yo code`, irá surgir a interface no terminal do gerador de extensões do Visual Studio Code.

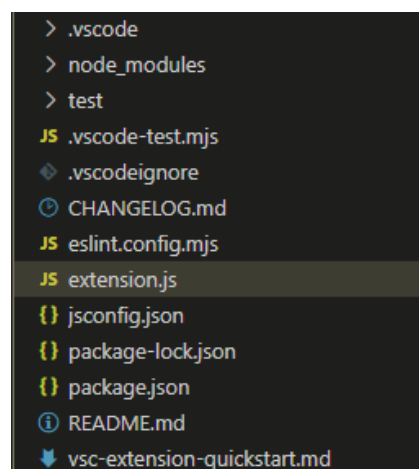


**Imagem 3. Gerador de Extensões do Visual Studio Code.**

Uma extensão para o VS Code pode ser programada em duas linguagens de programação, TypeScript e JavaScript, assim que a linguagem é escolhida resta adicionar o nome de sua extensão, a descrição e o pacote gerenciador da sua extensão. Após a conclusão, será criado um arquivo estruturado com o nome da extensão, abrindo a pasta com o VS Code será possível começar a programar a extensão.

## 5.2 Estrutura

A pasta criada pelo gerador de extensões do Visual Studio Code irá conter toda a estrutura necessária para programar a extensão, assim como é visível na Imagem 4.



**Imagem 4. Estrutura da Extensão.**

O arquivo `extension.js` é onde o programador coloca as funções de seus comandos para que sejam registrados e assim possam ser chamados quando o usuário que estiver usando a extensão precisar. A Imagem 5 mostra o Registro de Comandos.

```
function activate(context) {
  console.log('Congratulations, your extension "kata-kid" is now active!');

  // Comando para registrar usuário
  let registrarCommand = vscode.commands.registerCommand('extension.register', () => {
    register();
  });
  context.subscriptions.push(registrarCommand);

  // Comando para login
  let loginCommand = vscode.commands.registerCommand('extension.login', () => {
    login();
  });
  context.subscriptions.push(loginCommand);

  // Comando para logout
  let logoutCommand = vscode.commands.registerCommand('extension.logout', () => {
    logout();
  });
  context.subscriptions.push(logoutCommand);
}
```

Imagem 5. Registro de Comandos.

O arquivo `package.json`, vai servir para adicionar o comando que será chamado quando o usuário precisar, o comando terá um nome e a lógica para ser ativado quando o usuário desejar, assim como mostra a Imagem 6.

```
{
  "main": "./extension.js",
  "contributes": {
    "commands": [
      {
        "command": "extension.register",
        "title": "Registrar Usuário no Firebase"
      },
      {
        "command": "extension.login",
        "title": "Login no Firebase",
        "category": "Kata Kid"
      },
      {
        "command": "extension.logout",
        "title": "Logout do Firebase",
        "category": "Kata Kid"
      }
    ]
  }
}
```

Imagem 6. Nome do Comando e o comando.



A extensão funciona de maneira intuitiva, o usuário precisa chamar os comandos para que ela funcione, no Visual Studio Code pode-se usar (Ctrl+Shift+P) para que a aba de comandos apareça. Todos os requisitos funcionais são comandos que podem ser usados na extensão, assim como mostra a Imagem 7.

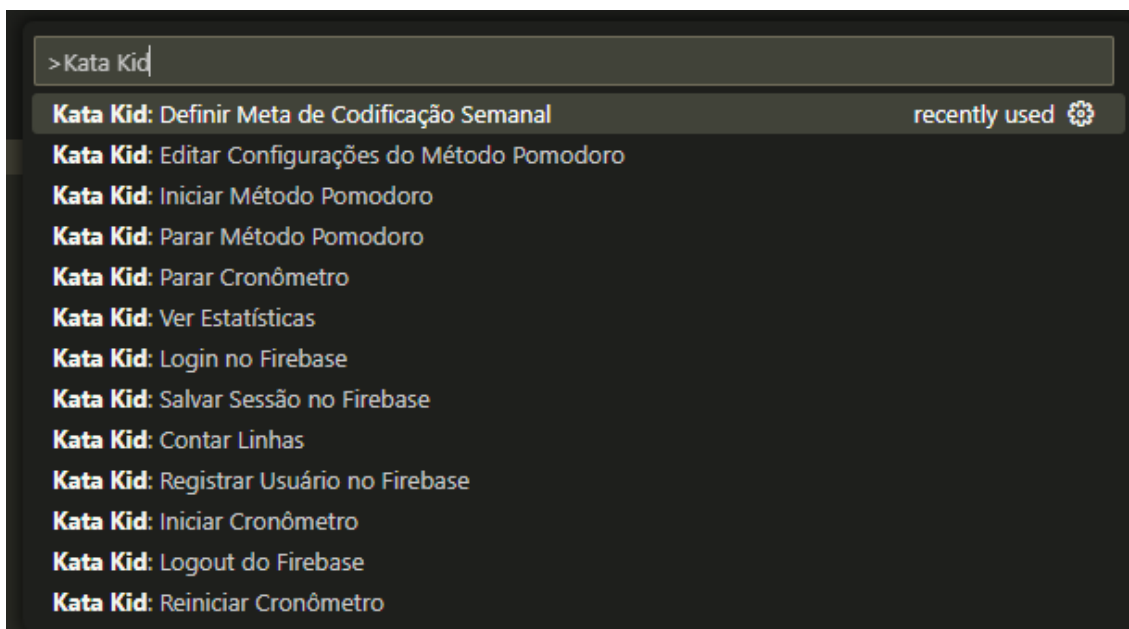


Imagem 7. Aba de Comandos.

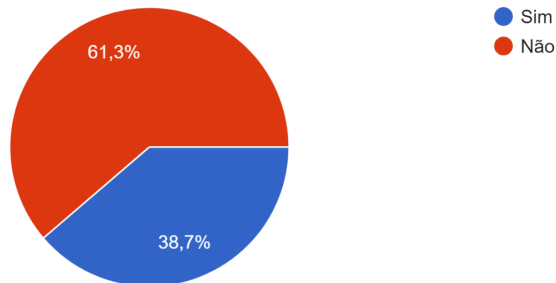
### 5.3 Pesquisa para levantamento de Requisitos

Com um entendimento básico sobre como criar uma extensão, é necessário saber o que a extensão irá fazer. Muitas ideias sobre o que a extensão iria fazer vieram de conversas com amigos e professores, além de ser baseada em outras extensões com o mesmo intuito. E com base em uma pesquisa com perguntas simples do modelo formulário feita com o Google Forms, com 31 participantes desenvolvedores. Foi possível juntar informações o suficiente para ter certeza que a ideia da extensão seria bem aceita.

Como podemos ver na Imagem 8, é perguntado se os desenvolvedores já utilizaram extensões para o Visual Studio Code, tendo um total de 61,3% dos participantes dizendo que não usam extensões. E outros 38,7% usam.

Você usa extensões no Visual Studio Code?

31 respostas

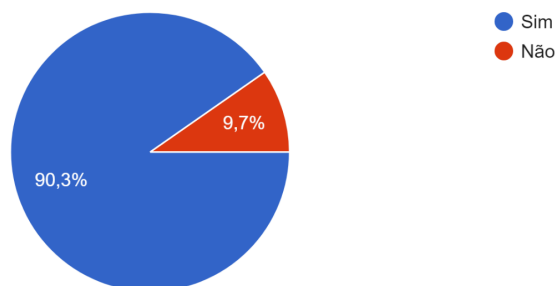


**Imagem 8. Primeira pergunta e respostas.**

A Imagem 9 mostra a segunda pergunta, que pede se os desenvolvedores usariam uma extensão que os ajuda a programar melhor. Com um total de 90,3% afirmando que usariam e outros 9,7% que não.

Você usaria uma extensão que te ajuda a programar melhor?

31 respostas

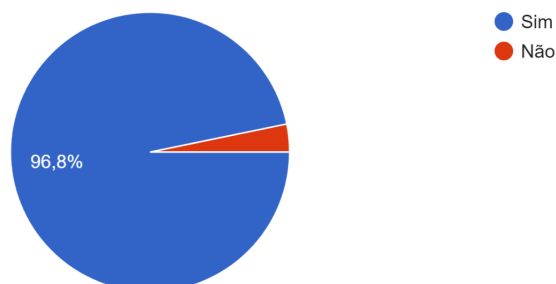


**Imagem 9. Segunda pergunta e respostas.**

A Imagem 10 mostra a terceira pergunta, que pede se os desenvolvedores usariam uma extensão com método de estudo implementado. 96,8% afirmaram que usariam, e 3,2% que não usariam.

Você usaria uma extensão que tem um método de estudos implementado a ela?

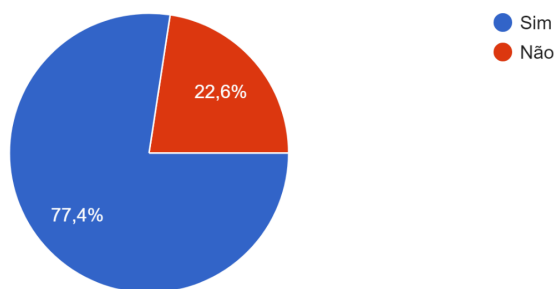
31 respostas



#### Imagem 10. Terceira pergunta e respostas.

A Imagem 11 mostra a quarta pergunta, que pede se os desenvolvedores usariam uma extensão que possui gerenciamento de tempo. 77,4% afirmaram que usariam, enquanto 22,6% não usariam.

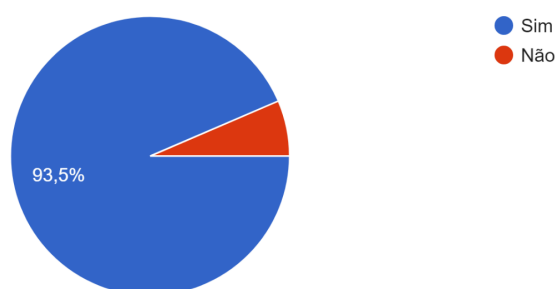
Você usaria uma extensão que tem gerenciamento de tempo?  
31 respostas



#### Imagem 11. Quarta pergunta e respostas.

A Imagem 12 mostra a quinta e última pergunta, que pede se os desenvolvedores usariam uma extensão que mostra estatísticas conforme o desenvolvimento do programador. A maioria afirma que sim, tendo um total de 93,5%. E 6,5% que não usariam.

Você usaria uma extensão que mostra estatísticas conforme o seu desenvolvimento?  
31 respostas



#### Imagem 12. Quinta pergunta e respostas.

### 5.4 Requisitos Funcionais

Com base nas respostas que foram obtidas pela pesquisa, podemos ver que uma extensão com gerenciamento de tempo, modo de estudo implementado e estatísticas de desenvolvimento foi bem aceita, com isso o levantamento dos requisitos funcionais e não funcionais para melhor entendimento de como irá funcionar a extensão foi feito.

De acordo com Chaim (2024), requisitos são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais. Requisitos funcionais delimitam a essência do que um sistema deve realizar. São descrições precisas das funcionalidades que o sistema irá oferecer, detalhando como ele deverá responder a entradas específicas e se comportar em diferentes cenários. Em outras palavras, os requisitos funcionais definem o "quê" o sistema faz. Já os requisitos não funcionais, por sua vez, estabelecem as condições e restrições sob as quais as funcionalidades do sistema serão entregues. Eles se referem a aspectos como desempenho, segurança, usabilidade, compatibilidade, entre outros. Enquanto os requisitos funcionais dizem respeito ao "quê", os não funcionais abordam o "como" o sistema deve operar. Ou seja, definem as características de qualidade e os padrões que o sistema deve atender.

**RF01. Registrar novo Usuário:** A extensão vai contar com registro de novos usuários para que seja possível salvar dados. Foi escolhido e-mail e senha, pois o Banco de Dados Firebase os utiliza para salvar a autenticação de usuários.

- a. E-mail: O usuário poderá fazer o registro usando um e-mail.
- b. Senha: O usuário terá que ter uma senha para que possa registrar-se com a extensão.

**RF02. Login:** Na extensão será possível fazer login para que o usuário sempre tenha seus dados salvos.

**RF03. Logout:** O usuário poderá fazer logout se desejar, caso queira sair de sua conta.

**RF04. Salvar Informações de Sessão:** O usuário por meio de comandos na extensão poderá salvar seus dados:

- a. O número de linhas em cada linguagem de programação usada;
- b. O tempo que passou programando;
- c. O tempo total e os ciclos que o usuário fez com o método Pomodoro.

**RF05. Iniciar e Pausar Cronômetro:** O usuário poderá iniciar e pausar o cronômetro com facilidade. Além de permitir que o usuário continue o exercício após a pausa, sem perder o tempo registrado.

**RF06. Reiniciar Cronômetro:** O usuário poderá reiniciar o cronômetro caso quiser.

**RF07. Tempo Total:** A extensão irá apresentar o tempo total gasto no exercício em formato claro e destacado.

**RF08. Contar Linhas:** A extensão pode contar todas as linhas de código que o programador fez, mesmo sendo de linguagens de programação diferentes, destacando-as (python, javascript, etc.).

**RF09. Método Pomodoro:** A extensão implementa o Método Pomodoro caso o usuário queira um melhor método para gerenciamento de tempo, este método dá ao usuário valores fixos já programados:

- a. Tempo de Ciclo: 25 minutos;
- b. Tempo de Descanso Curto: 5 minutos;

- c. Tempo de Descanso Longo: 20 minutos;
- d. Ciclos que são necessários antes de um descanso longo: 4 ciclos.

**RF10. Edição do Método Pomodoro:** A extensão implementa uma maneira do usuário editar como o Método Pomodoro irá funcionar, podendo:

- a. Mudar o tempo de cada Ciclo;
- b. Mudar o tempo de Descanso Curto;
- c. Mudar o tempo de Descanso Longo;
- d. Mudar quantos Ciclos são necessários para ter um descanso longo.

**RF11. Pausar Método Pomodoro:** O usuário será capaz de pausar o método Pomodoro.

**RF12. Metas:** O usuário pode definir metas para serem alcançadas em uma sessão de codificação, isto é, quantas linhas de código tem que ser escritas para atingir a meta.

**RF13. Sincronizar Dados:** Fazer a sincronização dos dados da extensão com o Firebase.

**RF14. Ver Estatísticas:** A extensão conta com uma função para o usuário poder ver as estatísticas de como está o seu desenvolvimento como programador. Assim que a função é chamada a extensão faz um GET no banco de dados, transfere os dados para um arquivo markdown que após é transformado em um HTML capaz de mostrar as estatísticas e dados do usuário.

- a. Ver quantas linhas de código já programou e de quais linguagens;
- b. Tempo gasto programando;
- c. Tempo com o Método Pomodoro e ciclos.

## 5.5 Requisitos Não-Funcionais

**RNF01. Manutenibilidade:** O código da extensão deve ser bem escrito e fácil de manter, para permitir futuras atualizações e aprimoramentos.

**RNF02. JavaScript:** Na criação da extensão será usada a linguagem de programação JavaScript.

**RNF03. Firebase:** Para banco de dados, foi utilizado o Firebase, para que o login dos usuários possa ser autenticado e assim também para salvarem seus dados de progresso.

- a. Firebase Authentication: Serve para fazer a autenticação de usuários que utilizarem a extensão. Para ser usado é preciso de:
  - i. E-mail;
  - ii. Senha.
- b. Firebase Realtime Database: Armazena e sincroniza os dados dos usuários.
  - i. Número de Linhas;
  - ii. Linguagem de Programação usada;
  - iii. Tempo Total do Cronômetro;
  - iv. Tempo Total do Método Pomodoro;
  - v. Ciclos do Método Pomodoro Concluídos.

## **5.6 Discussão dos Requisitos Funcionais e Não-Funcionais**

A extensão precisará de comandos do usuário para funcionar completamente. Pois é necessário chamá-los para que sejam ativados e assim fazerem o que o usuário deseja. Em trabalhos futuros pode ser feito atualizações para que os comandos sejam ativados automaticamente quando o usuário precisar.

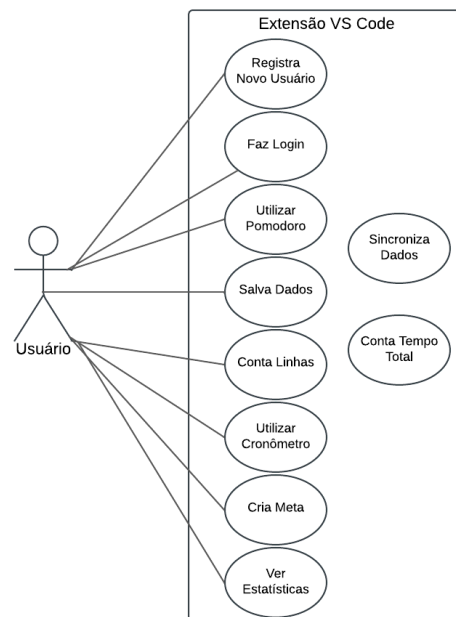
Fazer login no site será necessário caso o programador queira registrar seus dados e ver suas estatísticas. Sem o login a extensão não salva os dados no banco de dados, fazendo com que as estatísticas não possam ser vistas. Mas mesmo sem login o usuário poderá utilizar de outros comandos como “Definir Metas” e “Cronômetro”.

## **5.7 Diagramas UML**

Para fazer softwares ou projetos normalmente é utilizado os Diagramas UML (Unified Modeling Language) na qual de acordo com Guedes (2011) uma linguagem de modelagem visual amplamente utilizada na indústria de software para representar sistemas orientados a objetos. Ela oferece um conjunto de notações gráficas que permitem aos desenvolvedores modelar diversos aspectos de um sistema, desde os requisitos iniciais até a sua implementação final.

### **5.7.1 Diagramas de Caso de Uso**

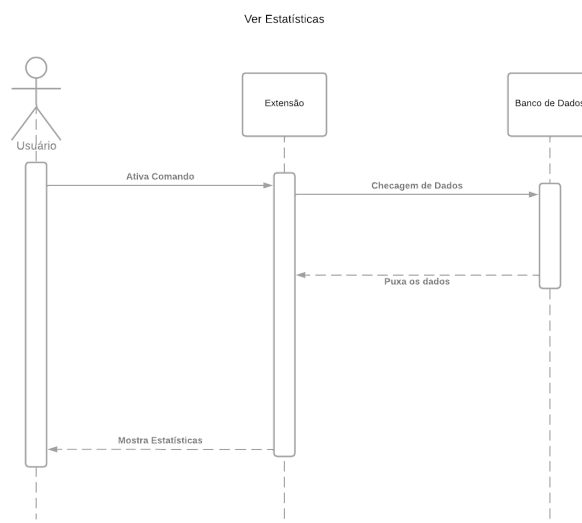
Antes de fazer a extensão foi feito o Diagrama de Casos de Uso de como seria a extensão. De acordo com Guedes (2011), diagramas de casos de uso tem como objetivo apresentar, de forma clara e concisa, as funcionalidades que um sistema deve oferecer aos seus usuários. Ele fornece uma visão externa do sistema, sem se aprofundar nos detalhes técnicos de sua implementação. As funcionalidades que estão ligadas aos usuários são funções que o próprio usuário precisa chamar. Assim como é mostrado na Imagem 13. A imagem contém dois usuários, mas os comandos do código podem realizar todas as ações que estão disponíveis, a não ser as que são feitas pela própria extensão como “Sincronizar Dados”.



**Imagem 13. Diagrama de Casos de Uso.**

### 5.7.2 Diagramas de Sequência

Após fazer o diagrama de Casos de Uso, foram feitos alguns diagramas de sequência para explicar melhor sobre a extensão. De acordo com Guedes (2011), o diagrama de sequência é um modelo comportamental que foca na ordem temporal das interações entre objetos em um processo. Ele identifica o evento inicial, o ator responsável e descreve como o processo ocorre por meio de trocas de mensagens entre os objetos, sendo geralmente baseado em casos de uso e diagramas de classes. Como podemos ver na Imagem 14 que mostra o que acontece quando o usuário ativa o comando de “Ver Estatísticas”.



**Imagem 14. Diagrama de Sequência - Ver Estatísticas.**

### 5.7.3 Modelo de Dados

Para salvar os dados foi necessário um banco de dados, para isso foi escolhido o Firebase e para entender como seria a estrutura foi feito um modelo de dados. De acordo com a IBM, a modelagem de dados é o processo de criar uma representação visual de um sistema de informação para mostrar conexões entre dados, seus tipos, relacionamentos, organização, formatos e atributos. Baseia-se em necessidades de negócios, com regras e requisitos definidos a partir do feedback das partes interessadas, servindo para projetar novos sistemas ou melhorar os existentes.

### 5.7.4 Entidades e seus Atributos

Usuário:

- ID (VARCHAR): Identificador único do usuário, usado para vinculá-lo ao Firebase Authentication.
- CiclosPomodoro (INT): Número de ciclos de Pomodoro realizados pelo usuário.
- date (VARCHAR): Data associada a salvamento de sessão do usuário.
- LinguagemMaisUsada (VARCHAR): Linguagem de programação mais utilizada pelo usuário.
- TempoTotal (INT): Tempo total gasto em atividades pelo usuário.
- TotalLinhas (INT): Número total de linhas de código escritas pelo usuário.
- TempoPomodoro (INT): Tempo dedicado especificamente aos ciclos de Pomodoro.

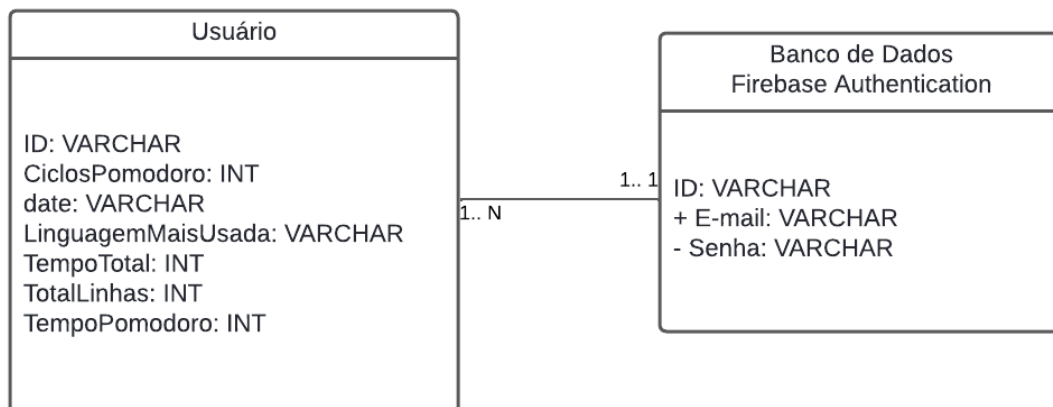
Banco de Dados Firebase Authentication:

- ID (VARCHAR): Identificador único no sistema de autenticação do Firebase (Vinculado ao ID do usuário na entidade Usuário).
- E-mail (VARCHAR): Endereço de e-mail do usuário, usado para autenticação.
- Senha (VARCHAR): Senha do usuário, armazenada de forma segura.

Existe uma ligação entre as entidades Usuário e Firebase Authentication por meio do atributo ID. Este relacionamento indica que o mesmo ID é usado tanto para armazenar dados no sistema principal quanto para autenticação no Firebase.

O modelo sugere que o Firebase Authentication é responsável por autenticar o usuário (via e-mail e senha) e a entidade Usuário armazena dados específicos do perfil e desempenho relacionados ao uso da extensão. Na Imagem 15 podemos ver o Diagrama de Entidade Relacionamento (ERD).





**Imagem 15. Diagrama de Entidade Relacionamento.**

## 5.8 Desenvolvimento

Para o funcionamento correto da extensão, foram necessárias serem usadas bibliotecas na qual são conjuntos de códigos pré-escritos que permitem resolver problemas de forma eficiente e simples [ESCOLA DNC, 2024]. Ao total no projeto foram instaladas 3 bibliotecas:

- Moment: é uma biblioteca que analisa, valida, manipula e exibe datas e horas em Javascript.
- Marked: uma biblioteca que permite converter Markdown em HTML. Markdown de acordo com Rosa (2023) é uma linguagem voltada para formatação de textos.
- Firebase: biblioteca usada para vincular o firebase com a extensão.

### 5.8.1 Cronômetro

No desenvolvimento do cronômetro além de ser um simples contador de tempo, foi adicionado atualizações, quando o usuário para por dois minutos, o cronômetro acaba parando de contar o tempo e retorna à contagem apenas quando o usuário volta a mexer no arquivo do código, além disso a cada 45 minutos o próprio cronômetro manda uma notificação para o programador fazer uma pausa e ter um tempo de descanso, isso foi pensado para o bem da saúde dos programadores. A Imagem 16 mostra a lógica do código. A Imagem 17 mostra o cronômetro em funcionamento e a Imagem 18 mostra a notificação quando o usuário fica inativo por dois minutos.

```

let codingTime = 0; // Tempo total codando, em segundos
let inactivityTime = 0; // Tempo de inatividade, em segundos
let timer = null; // Referência ao cronômetro
const INACTIVITY_LIMIT = 120; // 2 minutos de inatividade
const BREAK_REMINDER_INTERVAL = 45 * 60; // 45 minutos, em segundos
let statusBarItem = null; // Item da barra de status

function startCodingTimer() {
    if (timer) return; // Não inicia outro cronômetro se já estiver rodando

    if (!statusBarItem) {
        statusBarItem = vscode.window.createStatusBarItem(vscode.StatusBarAlignment.Right, 100);
        statusBarItem.show();
    }

    vscode.window.showInformationMessage("Cronômetro iniciado!");

    timer = setInterval(() => {
        codingTime++;
        inactivityTime++;
        statusBarItem.text = formatTime(codingTime);

        if (inactivityTime >= INACTIVITY_LIMIT) {
            stopCodingTimer();
            vscode.window.showWarningMessage("Cronômetro pausado devido à inatividade.");
        }

        // Verifica se é hora de lembrar o usuário de fazer uma pausa
        if (codingTime > 0 && codingTime % BREAK_REMINDER_INTERVAL === 0) {
            vscode.window.showInformationMessage(
                "Por que não fazer uma pausa né! Tome um café e relaxe por alguns minutos. ☕🕒"
            );
        }
    }, 1000); // Atualização a cada 1 segundo
}

function resetInactivityTimer() {
    inactivityTime = 0; // Reseta apenas o tempo de inatividade
}

```

Imagem 16. Comandos de inatividade e pausa.

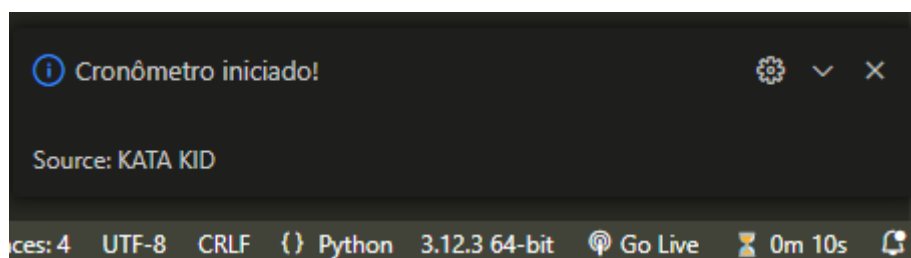


Imagem 17. Cronômetro em funcionamento.

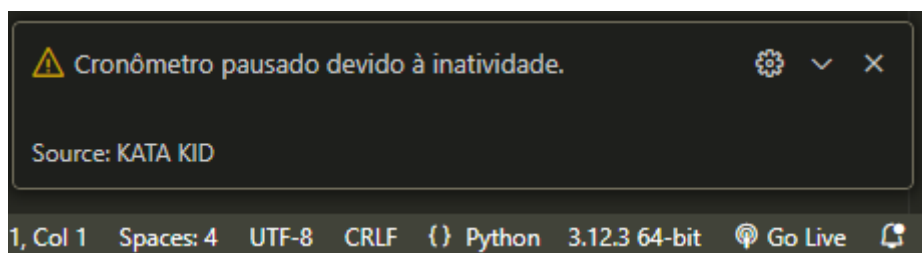


Imagem 18. Cronômetro pausado.

### 5.8.2 Comandos Login e Registro de Novo Usuário

Para que a extensão possa salvar os dados do programador no Banco de Dados Firebase, é necessário registrar-se com o Firebase Authentication, para que ele possa fazer a autenticação do usuário e assim salvar as estatísticas, seja ela, o tempo, número de linhas, entre outros, para isso foi feito uma sincronização com o Firebase. Quando o usuário estiver registrado e logado ele permite o usuário salvar e ver seus dados de desenvolvimento. A Imagem 19 mostra a lógica por trás do Login e Registro de Novos Usuários. A Imagem 20 e 21 mostra o funcionamento e a Imagem 22 a notificação após o registro de novo usuário.

```
let currentUser = null;

// Função de cadastro
function register() {
  vscode.window.showInputBox({ prompt: 'Digite seu e-mail para cadastro' }).then((email) => {
    if (!email) return;

    vscode.window.showInputBox({ prompt: 'Digite sua senha', password: true }).then((password) => {
      if (!password) return;

      // Usa a função createUserWithEmailAndPassword passando o auth
      createUserWithEmailAndPassword(auth, email, password)
        .then((userCredential) => {
          currentUser = userCredential.user;
          vscode.window.showInformationMessage(`Conta criada com sucesso! Bem-vindo, ${currentUser.email}`);
        })
        .catch((error) => {
          vscode.window.showErrorMessage(`Erro ao criar conta: ${error.message}`);
        });
    });
  });
}

// Função de login
function login() {
  vscode.window.showInputBox({ prompt: 'Digite seu e-mail' }).then((email) => {
    if (!email) return;

    vscode.window.showInputBox({ prompt: 'Digite sua senha', password: true }).then((password) => {
      if (!password) return;

      signInWithEmailAndPassword(auth, email, password) // Usa a função passando o auth
        .then((userCredential) => {
          currentUser = userCredential.user;
          vscode.window.showInformationMessage(`Login bem-sucedido! Olá, ${currentUser.email}`);
        })
        .catch((error) => {
          vscode.window.showErrorMessage(`Erro ao fazer login: ${error.message}`);
        });
    });
  });
}
```

Imagem 19. Código referente ao comando “Registro de Novo Usuário” e “Login”.

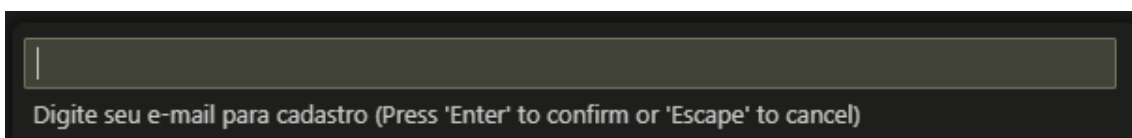
A screenshot of a VS Code input box. The input box is a dark gray rectangle with a light gray border. Inside the box, there is a vertical line cursor at the beginning. Below the input box, there is a text prompt in white: "Digite seu e-mail para cadastro (Press 'Enter' to confirm or 'Escape' to cancel)".

Imagem 20. Mensagem pedindo para que o usuário coloque e-mail para cadastro.

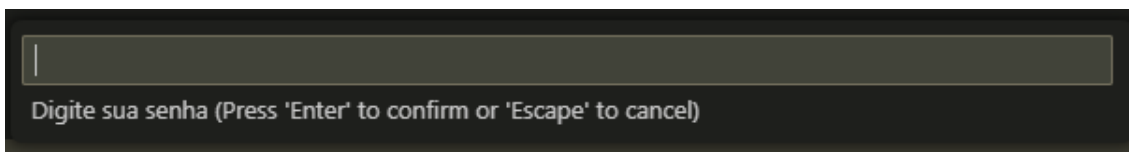


Imagem 21. Mensagem para o usuário colocar senha.

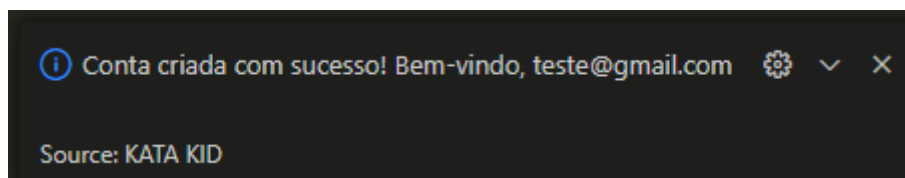


Imagem 22. Notificação após registro de novo usuário.

### 5.8.3 Edição do Método Pomodoro

A implementação do método Pomodoro contém as contagens de tempo inspiradas nas informações baseadas no livro de Cirillo (2013), mas se o usuário quisesse trocar o tempo para que fosse um ciclo mais longo o mesmo também pode ser feito, no código caso o usuário queira programar com o tempo de pomodoro normal, basta chamar o comando “Iniciar Método Pomodoro”, mas caso o usuário queira trocar a contagem de tempo, basta chamar o comando “Editar Método Pomodoro”. Após a ativação do comando, mensagens irão aparecer na aba de comandos, pedindo para que o usuário troque todas as contagens de tempo, além dos ciclos de pomodoro. Após a edição dos dados, eles ficam salvos em um Global State. No Visual Studio Code, o global state é uma funcionalidade do ExtensionContext, que permite armazenar informações de forma persistente no contexto de uma extensão [Microsoft, 2024b]. Na Imagem 23 pode-se visualizar uma das mensagens.

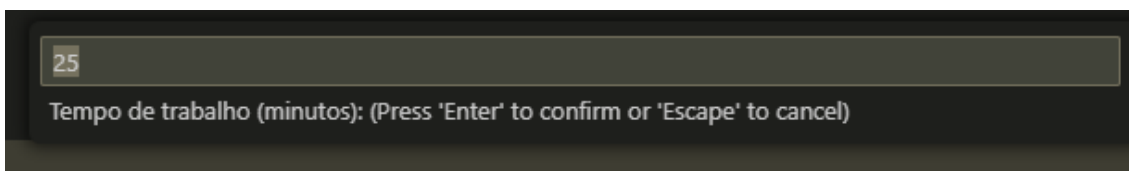


Imagem 23. Mensagem após a ativação do comando “Editar Método Pomodoro”.

### 5.8.4 Salvar Sessão

Quando o usuário está logado, ele é capaz de salvar sessão com o comando “Salvar Sessão”, após o chamado a extensão pega todos os dados armazenados até o momento da ativação do comando e salva tudo no banco de dados, como sessão de usuário. A Imagem 24 mostra a lógica por trás do código salvar sessão. E a Imagem 25 mostra a notificação de salvamento.

```
function saveSession() {
  const currentUser = getCurrentUser();

  if (!currentUser) {
    vscode.window.showErrorMessage("Usuário não está logado!");
    return;
  }

  const userId = currentUser.uid; // ID do usuário autenticado
  const date = new Date().toISOString(); // Formato ISO para ser facilmente interpretado

  const db = getDatabase(firebase);

  const codingStats = getCodingStats(); // Retorna um objeto com linhas por linguagem { 'Python': 100, 'JavaScript': 200 }
  const totallines = Object.values(codingStats).reduce((acc, lines) => acc + lines, 0);

  // Identifica a linguagem mais utilizada
  const mostUsedLanguage = Object.keys(codingStats).reduce((mostUsed, language) => {
    return codingStats[language] > (codingStats[mostUsed] || 0) ? language : mostUsed;
  }, '');

  const pomodoroStats = getPomodoroStats();
  const { totalPomodoroTime = 0, completedCycles = 0 } = pomodoroStats;

  const codingTimerStats = getCodingTimerStats();
  const { totalCodingTime = 0 } = codingTimerStats;

  const sessionData = {
    date: date,
    codingStats: codingStats, // Linhas de código em cada linguagem de programação
    totallines: totallines, // Total de Linhas
    mostUsedLanguage: mostUsedLanguage, // Linguagem mais utilizada
    totalCodingTime: totalCodingTime, // Total de Tempo
    totalPomodoroTime: totalPomodoroTime, // Total Tempo do Pomodoro
    completedPomodoroCycles: completedCycles, // Ciclos Completados
  };

  push(ref(db, `users/${userId}/sessions`), sessionData)
    .then(() => {
      vscode.window.showInformationMessage("Sessão salva com sucesso no banco de dados!");
    })
    .catch((error) => {
      vscode.window.showErrorMessage(`Erro ao salvar dados no banco: ${error.message}`);
    });
}
```

Imagem 24. Código referente ao comando “Salvar Sessão”.

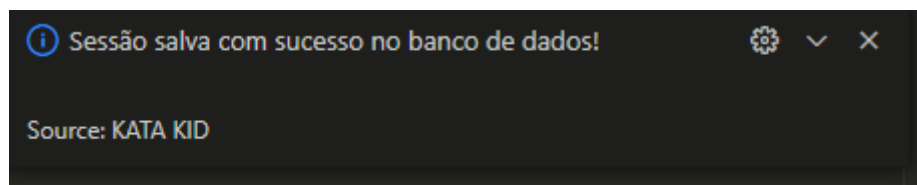


Imagem 25. Notificação referente ao comando “Salvar Sessão”.

### 5.8.5 Ver Estatísticas

Se o usuário estiver logado com a extensão ele poderá chamar o comando “Ver Estatísticas”. Quando chamado o código acessa os dados no banco de dados, pegando esses dados através do ID de usuário (Imagem 26) em seguida cria um arquivo markdown com os dados (Imagem 27) e depois os dados são lidos pelo HTML para que possam ser mostrados como estatísticas (Imagem 28), os dados são diário, semanal, mensal e total.

```
function getSessionData() {
  const currentUser = getCurrentUser();

  if (!currentUser) {
    vscode.window.showErrorMessage("Usuário não está logado!");
    return Promise.reject("Usuário não está logado.");
  }

  const userId = currentUser.uid;

  const db = getDatabase(firebase);
  const userSessionsRef = ref(db, `users/${userId}/sessions`);

  return get(userSessionsRef)
    .then(snapshot => {
      if (snapshot.exists()) {
        // Converte os dados em um formato utilizável
        const rawData = snapshot.val();
        const formattedData = Object.entries(rawData).reduce((acc, [key, value]) => {
          acc[value.date] = {
            totalLines: value.totalLines || 0,
            mostUsedLanguage: value.mostUsedLanguage || "Nenhuma",
            totalCodingTime: value.totalCodingTime || 0,
            totalPomodoroTime: value.totalPomodoroTime || 0,
            completedPomodoroCycles: value.completedPomodoroCycles || 0,
          };
          return acc;
        }, {});
        return formattedData;
      } else {
        vscode.window.showWarningMessage("Nenhuma sessão encontrada.");
        return {};
      }
    })
    .catch(error => {
      vscode.window.showErrorMessage(`Erro ao buscar dados: ${error.message}`);
      return Promise.reject(error);
    });
}
```

Imagem 26. Código referente ao comando “Ver Estatísticas”.

```
function generateMarkdown(data) {
  let markdown = `# Estatísticas de Codificação\n\n`;

  const dailyStats = {};
  const weeklyStats = {};
  const monthlyStats = {};

  // Organiza os dados em categorias
  Object.entries(data).forEach(([key, value]) => {
    const date = moment(key); // Interpreta datas ISO corretamente
    if (!date.isValid()) return; // Ignora datas inválidas

    const day = date.format('DD/MM/YYYY');
    const week = date.startOf('isoWeek').format('[Semana] WW');
    const month = date.format('MMMM'); // Nome do mês

    dailyStats[day] = dailyStats[day] || [];
    weeklyStats[week] = weeklyStats[week] || [];
    monthlyStats[month] = monthlyStats[month] || [];

    dailyStats[day].push(value);
    weeklyStats[week].push(value);
    monthlyStats[month].push(value);
  });

  // Função para calcular estatísticas acumuladas e linguagem mais usada
  const calculateAggregatedStats = (stats) => {
    const aggregated = {};
    Object.entries(stats).forEach(([key, values]) => {
      const totalLines = values.reduce((sum, session) => sum + (session.totalLines || 0), 0);
      const totalCodingTime = values.reduce((sum, session) => sum + (session.totalCodingTime || 0), 0);
      const totalPomodoroTime = values.reduce((sum, session) => sum + (session.totalPomodoroTime || 0), 0);
      const completedPomodoroCycles = values.reduce((sum, session) => sum + (session.completedPomodoroCycles || 0), 0);

      // Contagem de linhas por linguagem
      const languageLines = {};
      values.forEach(session => {
        if (session.codingStats) {
          Object.entries(session.codingStats).forEach(([Language, Lines]) => {
            languageLines[Language] = (languageLines[Language] || 0) + Lines;
          });
        }
      });
    });
  };
}
```

Imagem 27. Lógica ao comando “Ver Estatísticas”.

# Estatísticas de Codificação

## Hoje

Total de Linhas de Código: 0

Tempo de Codificação: 0 minutos

Tempo de Pomodoro: 0 minutos

Ciclos de Pomodoro Completos: 0

Linguagem Mais Utilizada: Nenhuma

Imagem 28. Estilo do HTML do comando “Ver Estatísticas”.

### 5.8.6 Definir Metas

O comando “Definir Meta de Codificação Semanal” não precisa que o usuário esteja logado para funcionar. Quando o usuário ativa este comando, poderá ser definido o número de linhas que o usuário precisa programar para atingir a meta. A Imagem 29 mostra a lógica por trás do comando. A Imagem 30 e 31 mostra as notificações que são enviadas ao usuário, por ter concluído a meta ou para mostrar quantas linhas de código faltam para concluí-la.

```
async function setCodingGoal(context) {
  try {
    // Solicita ao usuário para inserir a meta de linhas de código
    const goalInput = await vscode.window.showInputBox({
      prompt: 'Defina sua meta de linhas de código por semana',
      validateInput: (value) => {
        const parsedValue = parseInt(value, 10); // Converte para número
        if (isNaN(parsedValue) || parsedValue <= 0) {
          return 'Por favor, insira um número válido maior que zero.';
        }
        return null;
      },
    });
  });

  // Verifica se o usuário forneceu uma entrada válida
  if (goalInput) {
    const parsedGoal = parseInt(goalInput, 10); // Converte novamente para número inteiro
    if (isNaN(parsedGoal) || parsedGoal <= 0) {
      vscode.window.showErrorMessage('Meta inválida. Por favor, tente novamente.');      return;
    }

    // Salva a meta no globalState
    await context.globalState.update('codingGoal', parsedGoal);

    // Exibe uma mensagem de sucesso
    vscode.window.showInformationMessage(
      `Meta de ${parsedGoal} linhas de código por semana definida com sucesso!`
    );
  } else {
    vscode.window.showWarningMessage('Nenhuma meta foi definida.');  }
} catch (error) {
  console.error('Erro ao definir a meta de codificação:', error);
  vscode.window.showErrorMessage('Ocorreu um erro ao tentar salvar sua meta. Por favor, tente novamente.');
}

module.exports = { setCodingGoal };
```

Imagem 29. Lógica do comando “Definir Meta de Codificação Semanal”.

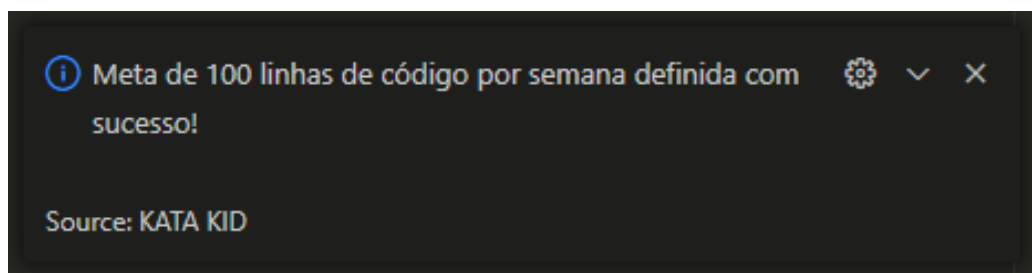
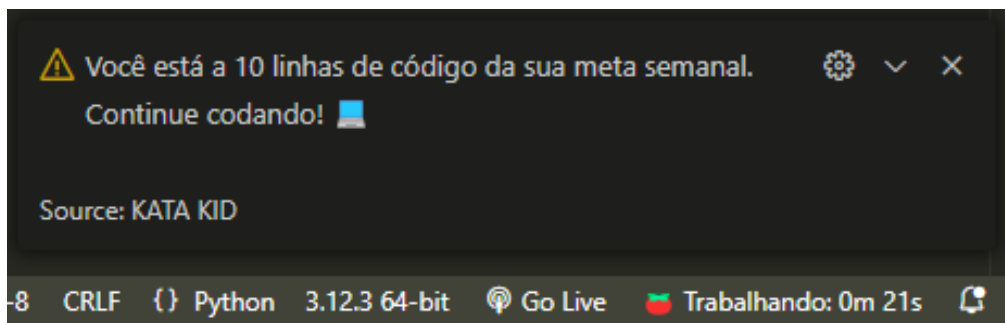


Imagem 30. Notificações do comando “Definir Meta de Codificação Semanal”.





**Imagem 31. Notificações do comando “Definir Meta de Codificação Semanal”.**

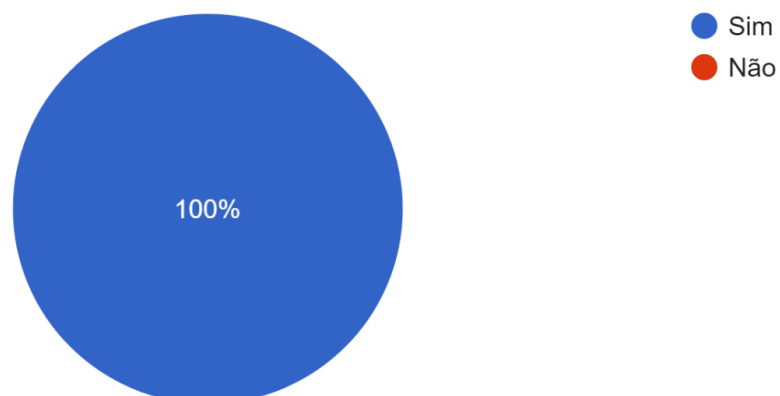
## **6. Considerações finais**

Para a conclusão do trabalho foi realizado um teste com 5 programadores que também realizaram o primeiro questionário, para que utilizassem a extensão e em seguida respondessem a outro questionário simples com 4 perguntas sobre a extensão. As perguntas foram feitas, para saber se a extensão está com um bom funcionamento, além de ideias para o futuro do projeto.

A primeira pergunta é para saber se a extensão possui alta usabilidade, como podemos ver na Imagem 32 todos acharam que tem sim usabilidade.

Você acha que a extensão possui alta usabilidade?

5 respostas



**Imagem 32. Primeira pergunta e respostas.**

A segunda pergunta é para avaliar a extensão no geral, ela possui 5 alternativas, sendo elas:

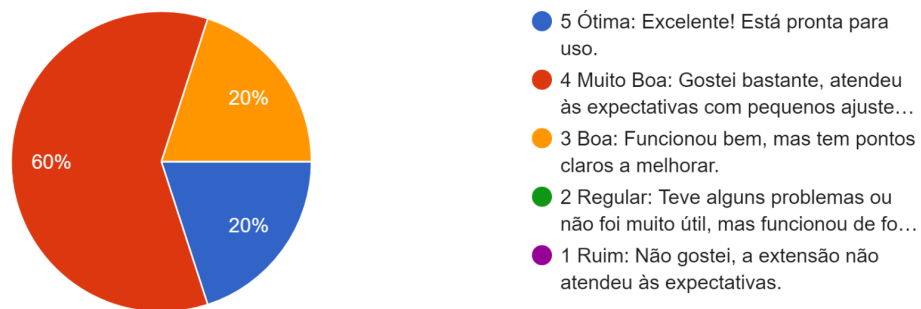
- Ótima: Excelente! Está pronta para uso.
- Muito Boa: Gostei bastante, atendeu às expectativas com pequenos ajustes a serem feitos.

- Boa: Funcionou bem, mas tem pontos claros a melhorar.
- Regular: Teve alguns problemas ou não foi muito útil, mas funcionou de forma básica.
- Ruim: Não gostei, a extensão não atendeu às expectativas.

Pelas respostas na Imagem 33, podemos ver que a extensão está funcionando bem, com as respostas acima de regular.

Em uma escala de 1 a 5, como você avaliaria a extensão no geral?

5 respostas

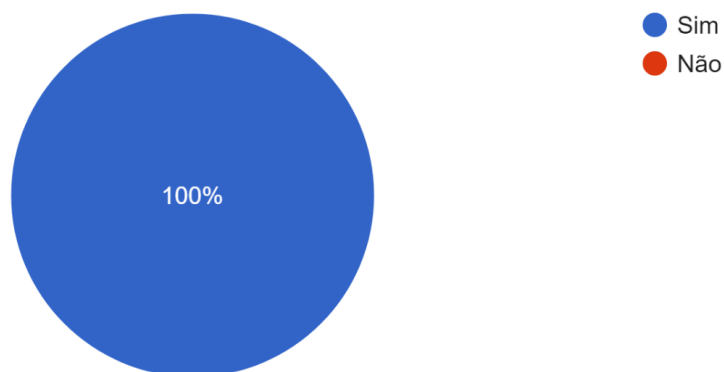


**Imagem 33. Segunda pergunta e respostas.**

A Imagem 34 mostra a terceira pergunta, que pede para os desenvolvedores se eles recomendariam a extensão para outros desenvolvedores que utilizam o Visual Studio Code. Todas as respostas foram positivas.

Você recomendaria essa extensão para outros desenvolvedores?

5 respostas



**Imagem 34. Terceira pergunta e respostas.**

A quarta e última pergunta, pede para os desenvolvedores sugerirem mudanças e melhorias para uma próxima versão da extensão. A Imagem 35 mostra as respostas.

Que melhorias você sugeriria para a próxima versão da extensão?

5 respostas

Contagem automática de linhas.

Salvar as linhas automaticamente.

Mais modos de estudo.

Lembrar do Login

Poderia ter mais tipos de metas.

**Imagem 35. Quarta pergunta e respostas.**

Com base nas respostas do questionário feito, conclui-se que a extensão tem boa usabilidade, um bom funcionamento e também atende aos requisitos estabelecidos.

A ideia principal era fazer um artigo sobre como fazer uma extensão para o Visual Studio Code, explicando o passo a passo de como criar uma extensão do zero, mas conforme o desenvolvimento continuava, a extensão começou a ter mais valor após a implementação de princípios do Método Pomodoro. Com isso, o trabalho que tinha o intuito de ajudar programadores iniciantes a fazer sua primeira extensão, se transformou em uma extensão que dá ao usuário um bom gerenciamento de tempo junto com um método de estudo muito eficaz para concentração e aprendizado, além do cronômetro que serve como marcador de tempo e as estatísticas do desenvolvimento dos usuários.

Para o futuro do projeto, espera-se que possam ser adicionadas mais informações para ajudar na criação de extensões, além de criar novas funções que ajudem o programador a sentir cada vez mais cativado ao programar como funções de gamificação, onde o usuário poderia ganhar conquistas e recompensas conforme for programando, criar uma rede virtual onde programadores possam compartilhar conquistas e estatísticas, e melhorar as funções que já estão no código.

Fazer este trabalho só foi possível devido ao conhecimento adquirido ao decorrer do curso e na hora de desenvolver a extensão foi possível colocar as habilidades em prática, seja com as pesquisas, com o planejamento dos requisitos, na criação dos diagramas, aperfeiçoar os conhecimentos com a linguagem de programação Javascript, além de melhorar a lógica de programação.

## Referências

- ANDRADE, António César de. **Fazendo sua própria extensão vscode**. 2020. Disponível em: <https://cibersistemas.pt/tecnologia/fazendo-sua-propria-extensao-vscode/#:~:text=O%20vscode%20já%20possui%20o,uso%2C%20vamos%2>. Acesso em: 16 jul. 2024.
- ANDRE LOUIS SOUZA RIBEIRO (Brasil). Alura. **O que é Firebase? Para que serve, principais característica e um Guia dessa ferramenta Google**. 2023. Disponível em: <https://www.alura.com.br/artigos/firebase?srsltid=AfmBOor0czmTWqYuXMKZ8W2sbgflduBXZKWtUyv2Y81BCOUTWpKwI4-k>. Acesso em: 28 nov. 2024.
- ARAUJO, Elias. **A Importância do Git e GitHub**. 2023. Disponível em: <https://www.dio.me/articles/a-importancia-do-git-e-github>. Acesso em: 22 jul. 2024.
- BAUERMAN, Gabriel. **Por que Git é tão difícil de aprender?** 2020. Quora. Disponível em: <https://pt.quora.com/Por-que-Git-é-tão-difícil-de-aprender>. Acesso em: 22 jul. 2024.
- CHAIM, Marcos Lordello. **ENGENHARIA DE REQUISITOS**: engenharia de sistemas de informação. ENGENHARIA DE SISTEMAS DE INFORMAÇÃO. Escola de Artes, Ciências e Humanidades | EACH | USP. Disponível em: [https://edisciplinas.usp.br/pluginfile.php/8001664/mod\\_resource/content/2/ACH2006\\_04\\_requisitos.pdf](https://edisciplinas.usp.br/pluginfile.php/8001664/mod_resource/content/2/ACH2006_04_requisitos.pdf). Acesso em: 28 nov. 2024.
- CIRILLO, Francesco. **The Pomodoro Technique**. 2013. Disponível em: [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwio2Y\\_A3rSKAxVUDrkGHS1JA9MQFnoECFgQAQ&url=https%3A%2F%2Fedisciplinas.usp.br%2Fpluginfile.php%2F4684649%2Fmod\\_folder%2Fcontent%2F0%2FPomodoroTechnique.pdf%3Fforcedownload%3D1&usg=AOvVaw361aKDOO3VXf-O4924Imi6&opi=89978449](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwio2Y_A3rSKAxVUDrkGHS1JA9MQFnoECFgQAQ&url=https%3A%2F%2Fedisciplinas.usp.br%2Fpluginfile.php%2F4684649%2Fmod_folder%2Fcontent%2F0%2FPomodoroTechnique.pdf%3Fforcedownload%3D1&usg=AOvVaw361aKDOO3VXf-O4924Imi6&opi=89978449). Acesso em: 16 nov. 2024.
- CNN (Brasil). **Novas tecnologias: tendências e o que esperar para o futuro**. 2023. Disponível em: <https://www.cnnbrasil.com.br/tecnologia/novas-tecnologias/>. Acesso em: 16 dez. 2024.
- DAGOBERTO HAJJAR (São Paulo/Sp - Brasil). **IMonitor IT (Brazil IT market survey) Estudo Trimestral do Mercado Brasileiro de TI**. 2024. ADVANCE Consulting. Disponível em: [https://www.advanceconsulting.com.br/pesquisa#:~:text=O%20mercado%20Brasileiro%20de%20tecnologia%20teve%20um,\(pessoas\)%20no%20Brasil%20independente%20do%20resultado%20local](https://www.advanceconsulting.com.br/pesquisa#:~:text=O%20mercado%20Brasileiro%20de%20tecnologia%20teve%20um,(pessoas)%20no%20Brasil%20independente%20do%20resultado%20local). Acesso em: 16 jul. 2024.
- DAHL, Ryan. **Download Node.js**. 2009. Disponível em: <https://nodejs.org/en/download/source-code>. Acesso em: 28 nov. 2024.

- ESCOLA DNC. (São Paulo, Brasil). **Bibliotecas em Node.js para o desenvolvimento de software.** Disponível em: <https://www.escoladnc.com.br/blog/a-importancia-das-bibliotecas-em-nodejs-para-de-senvolvimento-de-software/#:~:text=No%20contexto%20do%20Node.,populares%20de%20bibliotecas%20do%20Node>. Acesso em: 04 dez. 2024.
- GOOGLE. **Firestore Realtime Database.** 2014a. Disponível em: <https://firebase.google.com/docs/database?hl=pt-br>. Acesso em: 28 nov. 2024.
- GOOGLE. **Firestore Authentication.** 2014b. Disponível em: <https://firebase.google.com/docs/auth?hl=pt-br>. Acesso em: 28 nov. 2024.
- GUEDES, Gilleanes T. A.. **UML 2 - Uma Abordagem Prática.** 2011. Editora: Novatec Editora Ltda. Disponível em: <https://s3.novatec.com.br/capitulos/capitulo-9788575222812.pdf>. Acesso em: 03 dez. 2024.
- IBM. **O que é modelagem de dados?** Disponível em: <https://www.ibm.com/br-pt/topics/data-modeling>. Acesso em: 18 jan. 2025.
- INSPER (São Paulo/Sp - Brasil). **MERCADO DE TECNOLOGIA EM CONSTANTE EVOLUÇÃO: TENDÊNCIAS E OPORTUNIDADES DE CARREIRA.** 2023. Disponível em: <https://www.insper.edu.br/noticias/mercado-de-tecnologia/>. Acesso em: 18 jul. 2024.
- MICROSOFT. **Visual Studio Code.** 2024a. Disponível em: <https://code.visualstudio.com>. Acesso em: 28 nov. 2024.
- MICROSOFT. **Visual Studio Code: Common Capabilities.** 2024b. Disponível em: <https://code.visualstudio.com/api/extension-capabilities/common-capabilities>. Acesso em: 22 nov. 2024.
- OLIVEIRA, Jaime Bruno Cirne de. **Aprimorando a produtividade dos desenvolvedores com uma extensão para o Visual Studio Code Impulsionada por IA: uma abordagem com modelos de linguagem avançados.** 2024. Disponível em: <https://repositorio.ufrn.br/handle/123456789/57869>. Acesso em: 04 dez. 2024.
- ROSA, Giovanni Santa. **O que é markdown?** 2023. Disponível em: <https://tecnoblog.net/responde/o-que-e-markdown/>. Acesso em: 18 dez. 2024.
- SARAIVA, Márcia Denise Rodrigues Alves, Joana D’Arc Sampaio de Souza e Mallu Stephanie de Almeida Nunes. **TEMPO UNIVERSITÁRIO: FOCO NA JORNADA ACADÊMICA.** 2024. Disponível em: [https://www.editorarealize.com.br/editora/anais/conedu/2024/TRABALHO\\_COMPLETO\\_EV200\\_MD1\\_ID15215\\_TB5823\\_27102024194913.pdf](https://www.editorarealize.com.br/editora/anais/conedu/2024/TRABALHO_COMPLETO_EV200_MD1_ID15215_TB5823_27102024194913.pdf). Acesso em: 04 dez. 2024.
- VENDRAMETO, Barbara Dyna. **HEAR CODE.** 2022. Disponível em: [https://www.etecitapeva.com.br/arquivos/TCCs2022/TCC%20DS%20Prof%20Ana%](https://www.etecitapeva.com.br/arquivos/TCCs2022/TCC%20DS%20Prof%20Ana%20)

20Paula/TCC%20Des%20Sist%20Hear%20Code%20prof%20Ana%20Paula.pdf.  
Acesso em: 16 jul. 2024.

WAKATIME. 2025. Disponível em: <https://wakatime.com/vs-code>. Acesso em: 18 jan. 2025.