Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Desenvolvimento de um Guia de Princípios Éticos para Aplicações no Contexto de IA

Anayran Pinheiro de Azevedo

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Orientadora
Prof.a Dr.a Edna Dias Canedo

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Desenvolvimento de um Guia de Princípios Éticos para Aplicações no Contexto de IA

Anayran Pinheiro de Azevedo

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Prof.a Dr.a Edna Dias Canedo (Orientadora)
CIC/UnB

Prof. Dr.a Germana Menezes da Nobrega
CIC/UnB)

Prof.a Dr.a Fernanda Lima
CIC/UnB

Prof. Dr. Wilson Henrique Veneziano
Coordenador do Curso de Computação — Licenciatura

Brasília, 13 de maio de 2021

Dedicatória

Dedico esta obra a Deus e a todos os outros que, mesmo após tanto tempo, acreditaram que eu seria capaz de sair da Universidade de Brasília (UnB) com meu diploma em mãos. Vocês não tem ideia de como são importantes para mim, muito obrigado!

Agradecimentos

Agradeço em primeiro lugar a Deus, que por seu filho Jesus, não somente salvou a minha vida, mas também abençoou minha vida com todas as pessoas que vem a seguir. Agradeço à minha família, em especial, a minha mãe, Sônia, minha mãe do coração, Rosi, e minha avó, Dalice, que me deu a inspiração para iniciar este curso e trabalho e o incentivo para levar ele até o final. Agradeço à minha orientadora Edna Dias Canedo, pois sem as cobranças e o olhar atento ao meu trabalho não teria sido apresentado ao tema desenvolvido e ter conhecido um dos meu maiores companheiros até hoje.

Agradeço aos meus amigos, ao José Siqueira, pois sem ele não teria tido a ideia para realizar este trabalho e me apoiando desde o princípio a levar ele adiante. Aos meus amigos Daniel Ishy Torres Cavalcante e Helder Nozima, que não apenas me apresentaram a Jesus, mas também me deram os materiais e o apoio necessário para passar no vestibular e continuar no curso. Aos meus amigos Webysther e Natália, que me ajudaram demais na construção do trabalho prático. Ao meu time de Dance Games O2 Pump, em especial ao George, que me fez acreditar que eu chegaria ao final do curso. Um agradecimento muito especial ao meu amigo Alexandre Boutaud, pois sem sua ajuda não teria concluído várias etapas do curso, e seu apoio de 2012 até hoje foi essencial para persistir e insistir.

Muito obrigado a todos que me apoiaram a chegar até aqui.

Resumo

Em um mundo onde os avanços tecnológicos são tão rápidos, a tomada de decisão para uma ágil implantação e/ou adaptação se torna também necessária, levando à criação de várias metodologias que vão auxiliar desde a idealização, passando pelo planejamento e implantação, até a finalização destes projetos que foram pensados. Com este desafio se aplicando a todo o mundo, não seria impensável que mesmo no ramo de Inteligência Artificial o modelo de desenvolvimento de software ágil também estivesse incluso. Porém, neste modelo também vemos a necessidade de pensar nas implicações éticas do uso da tecnologia, que traz consigo importantes desafios a serem superados. Neste trabalho, é discutido como a aplicação da metodologia ágil impactou na criação de um guia para auxiliar os desenvolvedores e usuários finais na implantação de princípios éticos no contexto de aplicações de Inteligência Artificial (IA). Além disso, é apresentado como foi o desenvolvimento de um guia e seu devido uso através de um protótipo, utilizando o método ECCOLA criado por Vakkuri et al. [1], gerando assim um impacto imediato para os potenciais usuários. Mostramos também como modificar o guia para uso com outros parâmetros desenvolvidos, casos de uso e possibilidades de melhorias futuras, tanto nos projetos futuros quanto no projeto da própria ferramenta.

Palavras-chave: Inteligência Artificial, Ética em IA, Scrum, Planning Poker, Agile, Princípios em IA

Abstract

In a world where technological improvements are really quick, having the correct decision for a faster creation and/or adaption becomes more and more requested, asking for the creation of various methodologies which can help since the idealization, proceed through the planning and implementing until the end of those projects that were thought. With that challenge being applied for all the world, it would not be unthinkable that even when Artificial Intelligence area the agile software development were included. But, with this model we see how thinking on ethical implications using this technology becomes required to passthrough those challenges. On this work, we'll discuss how the agile methodology has impacted on the creation for a guide to help developers and stakeholders implementing those ethical principles at the Artificial Intelligence context. Besides that, it is presented how the development for this guide was proceeeded and using the ECCOLA methodology, created by Vakkuri et al. [1] as a prototype for this tool, generating an immediate impact for potential users. We also show how to modify this tool to use with other methodologies with other parameters, user cases and future improvements, both on future projects and on the development.

Keywords: Software Engineering, Artificial Intelligence, AI Ethics, Scrum, Planning Poker, Agile, AI Principles

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AI HLEG Artificial Inteligence High-Level Expert Group.

CSS Cascading Style Sheets.

EAD Ethically Aligned Design.

ER Engenharia de Requisitos.

HTML Hypertext Markup Language.

IA Inteligência Artificial.

IEEE Institute of Electrical and Electronics Engineers.

JS JavaScript.

OTAN Organização do Tratado do Atlântico do Norte.

UnB Universidade de Brasília.

XP Extreme Programming.

Capítulo 1

Introdução

Ao longo do tempo as soluções tecnológicas de tomada de decisões se tornam mais e mais cruciais, assim como o crescimento da área de Tecnologia da Informação e Comunicação (TIC) como um campo base para outras competências, desde a organização de estoques até a área de computação ubíqua, além do aumento acelerado do uso de dispositivos com alta capacidade computacional, como smartphones, tablets, smart TVs, smartwatches, assistentes pessoais, entre outros [2].

Esta dependência computacional atual tem se reforçado com a manifestação do uso de técnicas de computação baseadas em Inteligência Artificial (IA), como Aprendizado de Máquina (*Machine Learning*) e *Deep Learning* (DL) [8, 9]. Estas técnicas estão sendo empregadas de maneira amplamente difundida em áreas como vigilância, saúde, negócios, transportes, entre outros. Entretanto, este uso massivo de sistemas baseados em IA têm levantado questões acerca das discussões éticas que aquelas implicam [10, 9]. Incidentes relacionados amplamente divulgados em meios de comunicação chamaram a atenção da população em geral, e.g., um sistema baseado em IA utilizado como um juiz de um concurso de beleza que selecionou na sua maioria participantes brancos [11], e um chatbot da Microsoft que em pouco tempo apresentou características racistas [12].

Usualmente, ética em IA tem sido explorada na literatura em seu aspecto teórico, através de diretrizes e princípios éticos [13]. Apesar de necessárias, existem poucas explicações práticas para os desenvolvedores operacionalizarem estes princípios em seus projetos, o que é agravado pela necessidade de rapidez e lucro do mercado [13], no qual geralmente as considerações éticas envolvidas são um quesito de qualidade a ser considerada no software apenas após sua implantação [1]. Uma das maneiras de abordar esta urgência em entregas cada vez mais rápidas é através das técnicas de metodologias ágeis, que vêm sendo desenvolvidas desde a década de 70, e sendo formalmente compiladas em fevereiro de 2001 com o manifesto ágil [14].

Com o crescente uso desta abordagem, em projetos onde se envolve o desenvolvimento

de sistemas baseados em IA, os desenvolvedores enfrentam problemas, dilemas, dúvidas e discussões envolvendo a ética a ser abordada no projeto. A vasta maioria de ferramentas disponíveis para operacionalizar ética em IA são limitadas em termos de usabilidade e tem pouca evidência existe sobre testes, validade e impacto dos sistemas desenvolvidos [15].

Na Engenharia de Software, a fase de elicitação de requisitos é a primeira etapa do ciclo de desenvolvimento de software [2], e durante sua execução ocorre uma maior interação entre diferentes atores envolvidos no desenvolvimento do software e em seu uso, proporcionando um ambiente favorável para a discussão de questões éticas [16], além de haver uma prevenção de trabalho adicional ao serem consideradas as questões éticas nas fases iniciais do desenvolvimento de software, e não como um pensamento posterior [1].

O método ECCOLA, apresentado por Vakkuri et al. [1], é um método para operacionalizar a ética em IA na fase de elicitação de requisitos através do uso de cartas, semelhante ao Planning Poker, através de perguntas, aumentando a consciência ética em times de desenvolvimento ágil. De acordo o nosso conhecimento, não foi encontrado na literatura a implementação deste sistema, tampouco um caso de uso. Morley et al. [9] apontaram como oportunidades de pesquisa testar ferramentas ou métodos, identificar, propor ou melhorar os métodos e ferramentas de apoio à implementação dos princípios de ética no contexto de IA no processo de desenvolvimento de software. Nessa conjuntura, o objetivo deste trabalho é elaborar uma ferramenta para o auxílio da implementação de princípios éticos em IA por times de desenvolvimento de software, servindo como um guia e podendo ser customizado a diversas diretrizes ou contextos, e disponibilizá-lo publicamente.

1.1 Problema de Pesquisa

Existem na literatura trabalhos que apresentam coleções de ferramentas para a implementação de ética em IA [9, 17]. Apesar de auspiciosas, a grande maioria das ferramentas e métodos não possuem usabilidade, e oferecem pouca ajuda para serem colocadas em prática [10, 15, 9]. Além disso, há uma necessidade de colocar em prática as ferramentas, para encontrar o que funciona ou o que necessita ser melhorado. Um método avistado na literatura para operacionalizar ética em IA é o método ECCOLA[1]. Diante deste cenário, o nosso problema de pesquisa é que não existe, até então, uma implementação digital do método ECCOLA [1]. Portanto, este trabalho tem o objetivo de apresentar um software com a finalidade de operacionalizar ética em sistemas baseados em IA durante a fase de elicitação de requisitos com foco em times de desenvolvimento ágil, com o uso do método ECCOLA como protótipo.

1.2 Justificativa

A área de ética em IA surgiu como uma resposta às diversas consequências danosas aos indivíduos e à sociedade que o mau uso, mau projeto, abuso ou as não consequências intencionais que os sistemas baseados em IA podem causar [18]. Alguns dos danos potenciais causados por sistemas baseados em IA são: viés e discriminação, negação da autonomia individual, resultados não-transparentes, invasões de privacidade, insegurança, entre outros [18]. Além da ampla gama de diretrizes publicadas por diversos tipos de instituições a fim de mitigar estes problemas [19], também procura-se diminuir a lacuna entre a teoria e prática através de ferramentas. Entretanto, a existência destas ferramentas é considerada necessária, mas não suficiente [15]. Muitas são relativamente imaturas, possuem pouca usabilidade, e representam um trabalho adicional aos desenvolvedores ao serem colocados em prática por necessitarem modificações ao contexto [9, 10, 17, 1].

Algumas questões de pesquisa apresentadas por Morley et al. [9] são:

1. A criação de ferramentas que garantam às pessoas, como indivíduos, grupos e sociedades, uma oportunidade igual e significativa de participar na concepção de soluções algorítmicas em cada fase de desenvolvimento;
2. A avaliação dos instrumentos atualmente existentes para que se possa identificar o que funciona, o que pode ser melhorado, e o que precisa de ser desenvolvido;
3. O compromisso de reprodutibilidade, abertura e partilha de conhecimentos e soluções técnicas (por exemplo, software), também com vista a satisfazer (1) e apoiar (2).

No contexto em que os recursos são utilizados de forma mais racional, é necessária a criação de uma ferramenta que possa auxiliar os desenvolvedores na área de IA a adequarem também as questões éticas, uma vez que com o advento desta área se começa a entrar em uma zona cinza e ainda pouco explorada, onde as ferramentas existentes ainda são pouco efetivas [15].

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo deste trabalho é automatizar um método proposto na literatura para apoiar a elicitação de requisitos éticos em sistemas baseados em Inteligência Artificial no contexto de equipes de desenvolvimento ágil.

1.3.2 Objetivos Específicos

1. Compreender os princípios éticos para aplicações no contexto da Inteligência Artificial;
2. Automatizar o guia proposto para apoiar a elicitação de requisitos éticos no contexto de Inteligência Artificial.

1.4 Metodologia de Pesquisa

Ao longo deste trabalho, iremos utilizar a metodologia *Design Science Research* (DSR) [20]. Esta metodologia foi adotada pois auxilia a construção de um artefato e possibilita melhorá-lo através de um processo contínuo de refinamento e avaliação. Neste trabalho foi utilizado o ciclo de DSR baseado em Vaishnavi et al. [20], composto pelas seguintes fases: compreensão do problema, sugestão, desenvolvimento, avaliação e conclusão. Entretanto, iremos abordar neste trabalho apenas as fases de Compreensão do problema, Sugestão e Conclusão.

1. **Compreensão do problema:** Nesta fase ocorre a busca de informações sobre o problema a ser investigado, sem solucioná-lo todavia. Busca-se o entendimento e a descrição do problema, identificando os principais conceitos afetados pelo problema, além dos objetivos, causa do problema, efeitos e contribuições quando atingidos os objetivos. Portanto, nesta etapa foi conduzida uma exploração da literatura;
2. **Sugestão:** A sugestão é primordialmente uma etapa criativa no qual novas configurações são concebidas e assentadas em uma nova composição apropriada de novos elementos ou de elementos previamente existentes. Para esta fase, foi concebido um artefato, considerando o método ECCOLA [1]. O artefato proposto foi um guia – em forma de prova de conceito – para apoiar a implementação de ética em IA em equipes de desenvolvimento.
3. **Conclusão:** Por último, apresentamos as considerações finais, apontando possíveis trabalhos futuros.

As etapas de desenvolvimento e avaliação da ferramenta não foram estudados neste trabalho pelo fato de a construção do guia ser o principal foco a momento.

1.5 Estrutura do Trabalho

Este trabalho está organizado em quatro capítulos, além deste, consistindo em:

- **Capítulo 2:** apresenta a fundamentação teórica em relação aos conceitos de ética em IA. Além disso, são apresentados os trabalhos correlatos identificados na revisão de literatura.
- **Capítulo 3:** apresenta a proposta de um guia para apoiar a implementação da ética no desenvolvimento de aplicações no contexto de IA.
- **Capítulo 4:** apresenta as nossas considerações finais deste trabalho e trabalhos futuros.

Capítulo 2

Referencial Teórico

Neste Capítulo apresentamos a primeira etapa do *Design Science Research* – Compreensão do problema – onde apresentamos os conceitos necessários para o entendimento deste trabalho, passando pelos pontos da Engenharia de Software, Engenharia de Requisitos, elicitação de requisitos, Desenvolvimento Ágil de Software, Metodologia Scrum, *Planning Poker*, Inteligência Artificial, Ética em IA, princípios éticos e os trabalhos relacionados.

2.1 Engenharia de Software

A partir do início da década de 1940, os computadores foram projetados com um fim específico, como por exemplo o Colossus, primeira máquina computacional criada por Alan Turing que se tem registro, com o fim de decifrar mensagens criptografadas enviadas pelos alemães [21], durante a segunda guerra mundial. Esta era uma época na qual máquinas eram construídas para realizar apenas um único propósito computacional programável, com uma única tarefa a ser executada desde o princípio de sua existência. Pouco tempo após, vimos a criação dos primeiros computadores que permitiam executar mais do que um único programa, lendo a partir de cartões perfurados o que o computador deveria executar ou qual poderia ser uma resposta esperada, conforme descrito por Dijsktra [22].

Com o passar do tempo foi possível observar miniaturização, modernização e a popularização dos computadores, podendo estes serem usados dentro de firmas empresariais, órgãos governamentais, universidades e até mesmo em casas (apesar dos proibitivos preços), fazendo assim com que novas gamas de capacidades fossem dadas aos computadores. Com isto, naturalmente novos softwares e linguagens de programação foram sendo criados, trazendo consigo o início da popularização das áreas de desenvolvimento de software e como estas poderiam proporcionar um produto de qualidade com um tempo razoável e um custo factível. Porém, o que se via era exatamente o contrário, uma realidade na qual durante a produção destes aplicativos víamos projetos ultrapassando o limite de or-

çamento e o tempo previsto, uma qualidade bem aquém do esperado, não atendiam ao usuário final devidamente e de custosa manutenção e verificação dos códigos, conforme foi constado por Naur e Randell na Conferência do Comitê de Ciência da Organização do Tratado do Atlântico Norte (OTAN) [23]. A partir daí, verificando que alguns métodos derivados da administração poderiam funcionar também na produção de software, vemos o nascimento de métodos formais para o auxílio na construção de tais ferramentas de uso computacional. E daí, temos a criação e o desenvolvimento da área de Engenharia de Software, também citada pela primeira vez na conferência do comitê de ciência da OTAN [23].

Engenharia de Software é o processo que foca nas áreas de planejamento, desenvolvimento e entrega dos sistemas de software. É com este tipo de metodologia que conseguimos determinar de maneira mais apurada formas de como desenvolver o sistema, estimar prazos, recursos, e até mesmo espaços para melhora durante e após a conclusão do projeto. A Engenharia de Software nasceu da necessidade de entregar ao cliente uma maior garantia de qualidade, sem deixar de lado a preocupação com prazos, estes cada vez mais curtos perante as demandas que o mundo e sua evolução tecnológica tem exigido de acordo Sommerville[2]. Para Pressman [24], a Engenharia de Software envolve processos, métodos e ferramentas que permitem aos profissionais construir softwares bem desenvolvidos. Para Wazlawick [25], a Engenharia de Software é olhada como o meio de estudo, criação e otimização de processos para o desenvolvimento de software, onde as atividades essenciais de um engenheiro de software envolvem a especificação do software, seu desenvolvimento, validação e sua evolução.

Um dos motivos pelos quais a Engenharia de Software se tornou algo extremamente relevante é a crescente necessidade da sociedade e dos indivíduos de consumir e se relacionar com os sistemas de software, onde estes necessitam ser pensados e projetados com a maior excelência possível para que haja segurança, performance e possibilidade de manutenção. A partir de um olhar no aspecto econômico, um outro motivo é que, a longo prazo, se torna mais barato utilizar técnicas e métodos de Engenharia de Software ao invés de projetar sistemas que não possuam a padronização demandada sem estas técnicas e métodos, já que para a maior parte dos de sistemas de software, boa parte dos custos estão fortemente ligados às modificações realizadas no software após sua implantação, quando este já se encontra em produção [2].

Em suma, a Engenharia de Software é uma disciplina que envolve diversas áreas, a exemplo: processos de software, desenvolvimento ágil, engenharia de requisitos, testes de software, evolução de software e segurança. Assim, se dá a importância crucial dos sistemas de software para todos os aspectos da nossa sociedade atual.

2.1.1 Engenharia de Requisitos

A engenharia de requisitos (ou especificação de software) é a área responsável por entender e decidir quais serão as requisições necessárias ao sistema solicitado e verificar os impedimentos relacionados ao desenvolvimento e operação do sistema. Costuma ser um estágio crítico do processo de criação de um software, pois uma vez mal desenhado e com erros nesta fase, mais à frente no projeto, na implantação e manutenção do sistema problemas serão comuns de aparecerem, segundo Sommerville [2]. Para Pressman [24], é a área que engloba as tarefas e técnicas que guiam a uma noção mais precisa possível dos requisitos, sendo esta uma das ações principais da Engenharia de Software. Esta etapa existe com o propósito de criar uma documentação de requisitos acordados, onde cria a especificação de um sistema que possa atender às necessidades dos clientes atendidos para o desenvolvimento do software. Usualmente, os requisitos são postos para seus clientes e usuários finais em um nível mais alto, sem grandes detalhes técnicos, enquanto para os desenvolvedores é essencial que os detalhes técnicos sejam muito bem apresentados [2].

De acordo com Sommerville [2], são quatro níveis principais do processo de engenharia de requisitos, conforme apresentado na Figura 2.1, os quais são:

1. **Estudo de viabilidade**: Realiza-se uma aproximação a respeito da possibilidade de se cumprir com as requisições do usuário em questão, valendo-se de tecnologias contemporâneas, seja a nível de software ou de hardware. Esse estudo leva em conta se o sistema a ser criado será lucrativo sobre uma posição de negócio e se o mesmo pode ser construído sob condições financeiras limitadas. Um estudo de viabilidade deve ser algo sem grandes custos e de rápida conclusão, onde o resultado culmina na decisão ou não de continuar com a possibilidade de um estudo mais detalhado.

2. **Elicitação e análise de requisitos**: Esta é a parte do processo onde se inicia a formação mais aprofundada dos requisitos do sistema por meio da análise de outros sistemas existentes, além de debates com os potenciais clientes/usuários e compradores, verificação de tarefas e outras etapas. É nesta parte que a criação de protótipos ou modelos de sistemas é feita para ajudar a elucidar o sistema a ser criado.

3. **Especificação de requisitos**: A etapa de ação de tradução dos dados e informações coletados durante a etapa de análise de uma documentação que detalha o conjunto de requisitos. Geralmente, são dois os tipos de requisitos que podem ser inseridos nesta documentação. A parte onde o cliente menciona o que deseja criar para o seu produto de forma mais geral e sem a profundidade técnica, mais voltada para o objeto de desejo de uso é chamada de requisitos de usuário, enquanto a parte

onde há um maior detalhamento na descrição técnica a ser fornecido ao usuário é chamada de requisitos de sistema.

4. **Validação de requisitos**: Esta etapa é onde verificamos se o sistema está consistente, completo e de acordo com a realidade, onde não por acaso se descobrem os principais erros e a documentação deve ser alterada de forma a visar a correção destes problemas que foram mapeados durante esta etapa.

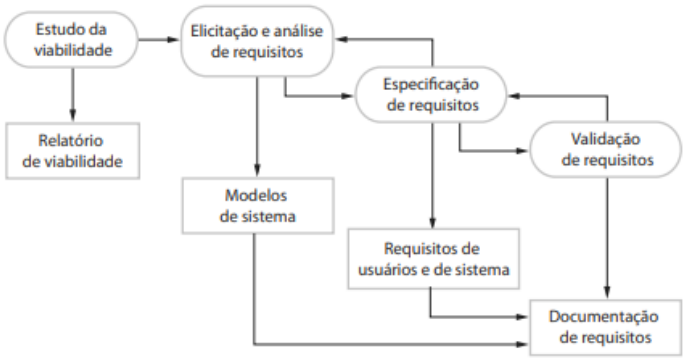

Figura 2.1: Processo de engenharia de requisitos [2]

De acordo com Sommerville [2], as atividades decorrentes do processo não são realizadas de forma linear e sequencial. Até alcançar a fase de entrega final, várias iterações novas são pensadas para novas definições, especificações e revisões dos requisitos, mostrando que tais etapas são de certa forma ligadas em cadeia. Nos modelos ágeis de desenvolvimento, como o *Extreme Programming*, os requisitos são criados, desenvolvidos e entregues de forma a incrementar sempre a iteração anterior de acordo as necessidades básicas e essenciais do usuário, enquanto o processo de elicitação de requisitos é feita pelos usuários que estão testando a ferramenta juntamente com o time de desenvolvedores.

Segundo Fleming [26], a área de engenharia de requisitos tem em mente três pontos em específico:

1. A área de desenvolvimento de requisitos do cliente fornece um conjunto de necessidades por parte do usuário que vão fornecer as informações para o desenvolvimento dos requisitos do produto;

2. A área de desenvolvimento de requisitos de produto fornece, a partir dos atributos dos elementos deste produto, informações sobre o melhor design a ser usado na otimização dos produtos ou de seus componentes;

3. A análise e a validação dos requisitos informações sobre a necessidade de análise do usuário, do produto e dos componentes do produto para definir, extrair e entender os requerimentos.

As práticas do terceiro ponto em si provêm suporte às práticas realizadas nos dois primeiros pontos. O processo associado com a área de engenharia de requisitos, e áreas correlatas, pode servir integralmente para a área técnica responsável, a qual pode interagir de forma incremental, contribuindo também na agilidade do desenvolvimento do produto para o usuário final.

2.1.2 Elicitação de Requisitos

Uma das primeiras atividades a serem realizadas nos processos de Engenharia de Requisitos (ER) é a elicitação de requisitos, onde ocorrem os processos de aprendizado, imersão e descoberta de necessidades [27]. Esta área está diretamente ligada às origens dos requisitos do sistema e à forma como os responsáveis pela Engenharia de Software conseguem coletá-los [28]. Para tanto, os profissionais da área, junto aos clientes e usuários finais do sistema em criação, trabalham de forma a obter insumos e dados a respeito do domínio da aplicação, dos serviços a serem ofertados pelo sistema, da maneira como ele se desempenha, de quais as limitações de hardware, e assim por diante, de acordo Sommerville [2].

Essa atividade é realizada de forma puramente humana, onde os *stakeholders* são identificados e, a partir daí, o relacionamento entre equipe de desenvolvimento e cliente é consolidado [28]. Essa atividade passa pelo envolvimento de diversos perfis de indivíduos dentro de uma organização, como por exemplo os próprios *stakeholders*, porém também se observam os usuários finais que irão operar o sistema e/ou qualquer outra pessoa que seja afetada com este uso, de acordo Sommerville [2].

A elicitação tem como focos a identificação do problema, proposição de elementos para a solução, a negociação de abordagens distintas e a especificação de um conjunto primitivo de requisitos do produto em um ambiente que seja apto para o alcance deste objetivo [24]. Um dos aspectos principais para uma elicitação de requisitos ser alcançada com sucesso é a fácil e eficiente comunicação entre os diversos *stakeholders* [28].

Para tanto, é necessária uma coleção de modelos que seja consistente nos mais variados níveis de abstração, visando facilitar a comunicação entre os usuários do sistema (*stakeholders*) e os engenheiros de software [28]. Temos à disposição vários tipos de modelos de elicitação e análise de requisitos, onde os quais apresentam, para cada organização, a sua versão ou instância que depende de fatores ambientais locais, como por exemplo o nível de conhecimento dos usuários, a espécie de sistema a ser criado, quais normas serão usadas, entre outros [2].

2.2 Desenvolvimento Ágil de Software

Idealizado em 2001, em uma carta pública divulgada na internet, o Manifesto Ágil, criado por Kent et al. [14], veio para se traduzir em uma revolução na forma de transformar os projetos para criação de produtos baseados em programação. Com o lema de "valorizar indivíduos e interações, software em funcionamento, colaboração com o cliente e responder às mudanças", a mudança de paradigma que o manifesto trouxe na produção de software foi imediata, tendo esses princípios surgido derivados a partir do modelo *Extreme Programming*, também conhecido como XP. Com esta nova forma de desenvolvimento, o foco deixa de ser um modelo de construção de software em cascata, sem interações e respostas rápidas e passa para um modelo muito mais adaptável e maleável de acordo as necessidades de negócio. Para Sommerville [2], tais características do desenvolvimento ágil de software passava pelos seguintes princípios:

- **Envolvimento do cliente**: O cliente precisa estar a par do que se sucede durante o desenvolvimento de seu produto, fazendo uma constante verificação da criação do resultado de seu sistema, além de poder sugerir implementações e melhorias durante este processo.
- **Entrega incremental**: O desenvolvimento do produto é feito de forma incremental e individual junto ao cliente.
- **Pessoas, não processos**: As equipes devem ter suas habilidades de desenvolvimento do produto reconhecidas ter permitido o seu desenvolvimento com liberdade, oportunizando assim que os membros sejam respeitados em suas peculiaridades e atuem sem manuais pré-definidos.
- **Aceitar as mudanças**: É necessário ter em foco que os requisitos do sistema irão mudar com o tempo de desenvolvimento, por isso todo o processo deve ser realizado permitindo a alteração e acomodação destas alterações não previstas.
- **Manter a simplicidade**: Manter sempre em vista o lado simples das coisas, tanto do software a ser criado quanto dos processos de desenvolvimento deste programa, trabalhando de forma proativa para eliminar possíveis complexidades que o sistema venha a apresentar quando possível.

2.2.1 Metodologia Scrum

Scrum é uma metodologia ágil para gestão e planejamento de projetos de software. Um framework dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos [29]. De acordo Schwaber e Sutherland [5], mesmo sendo uma metodologia fácil de

entender e de baixos recursos, é um método de difícil execução, por ser difícil de dominar. Segundo Pressman [24], o entendimento a forma de lidar do Scrum batem diretamente com o Manifesto Ágil. A metodologia Scrum é formada por times, papéis, eventos, artefatos e regras, conforme apresentando na Tabela 2.1.

Cada um dos elementos nesta metodologia tem um foco específico e todos são cruciais para sucesso no uso. Schwaber e Sutherland [5] descrevem os principais pilares que são utilizados pela metodologia Scrum: Transparência; Inspeção; Adaptação. De acordo com o time os membros aprendem e exploram os valores supracitados à medida em que lidam com os projetos usando a metodologia. É possível perceber que os valores de comprometimento, coragem, foco, transparência e respeito são abraçados e se tornam parte comum no time. Conforme estes valores são internalizados e há mais vivência com eles, as chances de sucesso no uso da metodologia [5] aumentam.

Tabela 2.1: Time, artefatos e eventos do *Scrum* [5]

Papel	**Descrição**
Product Owner	Profissional responsável por entregar o máximo possível de valor ao time de desenvolvimento e ao produto.
Time de Desenvolvimento	Equipe de técnicos responsáveis pela entrega de uma versão utilizável do produto a cada *Sprint*.
Scrum Master	Profissional responsável pela fixação da teoria, práticas e regras do *Scrum* por garantir que esta metodologia esteja em conformidade.
Artefato	**Descrição**
Backlog do Produto	Descrição de tudo o que o produto deve ter, estando este em constante evolução e sempre fornecendo históricos de referência para as mudanças a serem registradas no produto.
Backlog da *Sprint*	É composta pelos itens da Product Backlog selecionados para aquela *Sprint*, mais o plano de como entregar o incremento do produto e atingir os objetivos da *Sprint*. Os itens são analisados pela equipe de desenvolvimento e divididos em tarefas.
Evento	**Descrição**
Sprint	Parte central do *Scrum* que ocorre por um tempo de duas a quatro semanas, período onde uma parte do produto é criada.
Reunião de Planejamento de *Sprint*	Evento no qual o trabalho a ser realizado na *Sprint* é planejado por todo o Time Scrum.
Reunião Diária	Reunião com duração máxima de 15 minutos, para que o Time de Desenvolvimento possa informar o que foi realizado no trabalho no dia anterior, o planejamento do dia corrente e os possíveis bloqueios que estejam ocorrendo para o progresso da *Sprint*.
Revisão da *Sprint*	Reunião realizada ao final da *Sprint* para revisar o incremento e adaptar o Backlog do Produto, se necessário.
Retrospectiva da *Sprint*	É uma oportunidade para o Time Scrum inspecionar a si próprio e criar um plano de melhorias a serem aplicadas na próxima Sprint.

No Scrum, as funcionalidades a serem implementadas em um projeto são mantidas no Backlog do Produto. O processo de desenvolvimento ocorre de forma iterativa, e a cada iteração se dá o nome de *Sprint*, a qual possui um prazo de duas a quatro semanas para a sua conclusão. A cada *Sprint*, se realiza uma Reunião de Planejamento de *Sprint*, um evento onde o *Product Owner* prioriza os itens do Backlog do Produto e a Equipe de Desenvolvimento separa as atividades que ela será capaz de implementar durante a *Sprint* que se inicia. As atividades selecionadas para a *Sprint* são enviadas do Backlog do Produto para o Backlog da *Sprint*. Além disso, a cada dia a Equipe de Desenvolvimento faz uma reunião de breve duração (geralmente até 15 minutos) com o intuito de alinhar o conhecimento dos membros da equipe sobre o trabalho que está sendo desenvolvido, além de realizar o planejamento para o dia e discutir possíveis bloqueios que possam estar sucedendo durante a execução dos trabalhos. Ao final de uma *Sprint*, a equipe realiza uma Revisão da *Sprint*, onde são mostradas as funcionalidades implementadas ou o que pode ter falhado durante a entrega. No fim, é realizada uma Retrospectiva da *Sprint* onde a equipe avalia o que foi realizado, o que faltou realizar e faz o planejamento da futura *Sprint* [5].

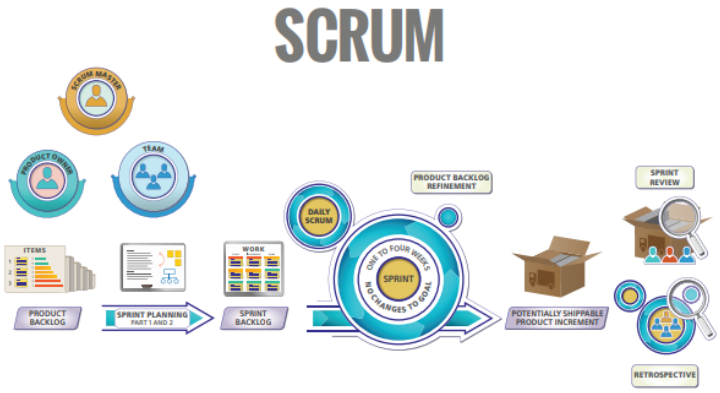

Figura 2.2: Como uma *Sprint* é planejada, executada e seus membros [3].

2.2.2 *Planning Poker*

O *Planning Poker* consiste em uma técnica de estimativa lúdica e de fácil execução voltada a metodologias ágeis, com o foco em tamanho de complexidade da história para todos os membros de um time ágil [30, 31, 32]. Este modelo é voltado para que cada membro possa colocar em vista, durante a reunião de planejamento do projeto a ser executado enxergam e o que fariam com aquela história [5]. Durante este processo, a equipe de desenvolvimento e os especialistas do negócio (geralmente o *Product Owner*), juntamente com um *Scrum Master*, se reúnem para chegarem a um consenso relacionado às complexidades

dos trabalhos a serem executados [33]. Geralmente, este processo tende a criar resultados mais aceitáveis em relação a risco e erros nas estimativas por reduzir ambos os problemas [34].

Neste processo, os participantes da reunião são providos de um *deck* de cartas para o *Planning Poker*. Cada carta possui um valor único, como por exemplo 0, 1/2, 1, 2, 3, 5, 8, 13, 21 e "?", este último geralmente utilizado quando o membro não sabe estimar o valor e deixa em aberto para a discussão e definição na rodada seguinte. Outra série comumente utilizada é a série de Fibonacci para os valores das cartas. Não existe um consenso a respeito tais valores, porém a sequência de Fibonacci é a mais comum a ser utilizada [35].

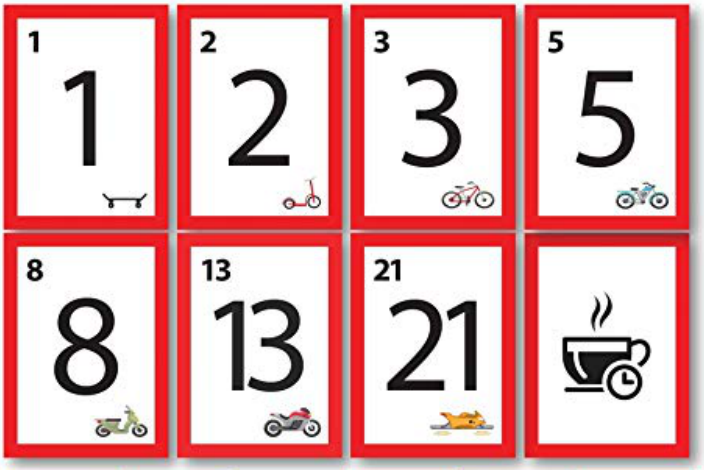

Figura 2.3: Exemplo de cartas de um *Planning Poker* [4].

Os processos e os passos para o *Planning Poker* são simples e de fácil aplicação [34]. Primeiramente, uma história de usuário será escolhida pelo moderador do jogo. Depois, o representante do cliente, ou o *Product Owner*, explica de forma clara, para que todos os envolvidos absorvam a história com atenção. Em caso de qualquer questionamento, o *Product Owner* é o responsável pela resposta. Logo após, cada participante estima o tamanho da história mostrando a carta que indique um valor.

Este valor deve ser levando em consideração fatores que possam influenciar o projeto e o produto, como a complexidade e o risco de negócio envolvendo a história, por exemplo. Caso haja unanimidade no valor das cartas apresentadas pelos membros envolvidos, o valor é dado à história e o time escolhe outra história para a estimativa. Caso contrário, todos os membros do *Planning Poker* que escolheram o maior e o menor valor precisam explicar os motivos de terem dado aquele valor para a história, e defenderem sua sugestão de valor. Após este ponto, o time conversa entre si sobre a história e discute a respeito de suas especificações, requerimentos e limitações em detalhe. Imediatamente após a discussão, a estimativa deve ser reavaliada. Este processo deve ser feito até que o time

chegue a um consenso no valor da história. Após o consenso, este procedimento deve ser repetido para todas as histórias que fazem parte daquela reunião de planejamento e, assim, passarem a fazer parte da *Sprint* corrente. [36].

De acordo com Gandomani et al. [37], após estudo em uma empresa anônima, foi verificado que para o modelo funcionar é necessário maturidade da equipe como um todo, desde o processo de estimativa até o julgamento final do grupo para estimar corretamente a complexidade da história. Nada impede que em algumas horas o time chegue a um consenso, sem longas conversas, acreditando no que um dos membros tem a relatar sobre aquela história, ou que em determinadas circunstâncias as longas conversas sejam necessárias para, assim, se chegar a uma unanimidade. De acordo com o autor, este segundo ponto costuma trazer maior precisão na estimativa da complexidade do projeto. Porém, para alcançar tal ponto, é necessário que a equipe trabalhe em regime colaborativo, permitindo que o consenso venha de forma mais natural e precisa.

2.3 Inteligência Artificial

Quando abordamos o assunto "inteligência artificial", temos uma complicada missão para a definição deste conceito, porém ao longo do tempo este assunto se dividiu entre quatro linhas de raciocínio:

1. Sistemas que pensam como seres humanos: "O novo e interessante esforço para fazer os computadores pensarem... máquinas com mentes, no sentido total e literal", de acordo com Haugeland [38].
2. Sistemas que atuam como seres humanos: "A arte de criar máquinas que executam funções que exigem inteligência quando executadas por pessoas", de acordo com Stalder [39].
3. Sistemas que pensam racionalmente: "O estudo das faculdades mentais pelo seu uso de modelos computacionais.", de acordo com Charniak [40].
4. Sistemas que atuam racionalmente: "A Inteligência Computacional é o estudo do projeto de agentes inteligentes.", de acordo com Poole et al. [41].

Geralmente, as linhas de pensamento 1 e 3 apontam para o processo de pensamento e raciocínio, enquanto as 2 e 4 ao comportamento. Além disso, as linhas de pensamento 1 e 2 aferem o sucesso em razão à fidelidade relacionada ao desempenho humano, enquanto nas 3 e 4 aferem o sucesso fazendo uma comparação a um conceito ideal de inteligência, que se chamará de racionalidade. Um sistema é racional, inteligente, se "executa tudo de acordo" com os dados fornecidos a ele, segundo Russel e Norvig [6].

Historicamente, as quatro perspectivas para o estudo da inteligência artificial têm sido seguidas. Sem surpresas, há uma tensão entre abordagens com o foco ao redor de seres humanos e abordagens focadas na racionalidade. Uma abordagem focada em seres humanos precisa ser feita com o empirismo, envolvendo hipóteses e afirmação através de experimentos científicos. Enquanto isso, a abordagem racionalista é um misto de matemática e engenharia [6].

A IA ainda é um assunto de difícil complexidade dependendo do tema ao qual é atrelada a sua abordagem. Ainda não sabemos se o homem seria capaz de criar a real inteligência artificial, ou chegar perto de construir um simulacro que possa reproduzir as bases de como o cérebro humano se comporta, que é a base sua criação. Hoje, o que temos são noções de que seus conceitos desenvolvidos ao longo de anos trazem incontáveis progressos e benesses para humanidade, e que de certa forma ela sempre inovará e evoluirá gradativa e progressivamente [42].

Segundo Russel e Norvig [6], suas bases encontram-se em distintos campos, a partir de distintas perguntas, perspectivas e desafios, conforme apresentado na Tabela 2.2.

Tabela 2.2: Relação campo versus pergunta [6]

Filosofia	Podem ser utilizadas regras formais para tirar conclusões válidas? Como a mente funciona em um cérebro físico? De onde vem o conhecimento? Como o conhecimento leva à ação?
Matemática	Quais são as regras formais para tirar conclusões válidas? O que pode ser computado? Como raciocinamos com informações incertas?
Economia	Como devemos tomar decisões para maximizar o retorno? Como devemos fazer isso quando outros não podem ir junto? Como devemos fazer isso quando a recompensa pode estar longe no futuro?
Neurociência	Como os cérebros processam informações?
Psicologia	Como os seres humanos e os animais pensam e agem?
Engenharia e Computação	Como podemos construir um computador eficiente??
Teorias de Controle e Cibernética	Como as maquinas podem operar sob seu próprio controle?
Linguística	Como a linguagem se relaciona com o pensamento?

De forma geral, para Kaufman [43], e em um olhar simplificado, podemos pensar a IA como uma reexecução dos comportamentos que o cérebro humano pode controlar. A ação de dançar, por exemplo, é controlada pelo cérebro (acompanhar o par, coordenação da movimentação no ritmo de partes do corpo em geral), assim como o respirar é igualmente controlado pelo cérebro, mesmo que de forma indireta: as sensações que chegam no cérebro são parte do domínio da inteligência, estando potencialmente no campo da IA. Porém, o "processo mental" da IA supera o processo mental humano, se, por um lado, a IA executa tarefas que são por suposto competências dos seres humanos, por outro lado sua

capacidade ultrapassa limitações humanas como observado em distintas situações do dia a dia com o uso das tecnologias apropriadas, como por exemplo em um jogo de xadrez [44] ou Go [45].

2.4 Ética em IA

A palavra ética, tentando simplificar uma coisa que de forma alguma pode ser chamada de simples, indica uma noção de comportamento adquirida ou conquistada, portanto, não-instintiva, de acordo Kaufman [43]. Trata-se, de toda forma, de um modo de um ser humano agir, construído socialmente com base nas relações e na interação entre os seres humanos. Refere-se diretamente à natureza da ação humana. Se a IA representa uma nova espécie inteligente coexistindo com a espécie humana, nos parece legítimo supor que precisamos de uma nova ética. Este nos parece um dos maiores desafios postos no campo das ciências sociais, envolvendo a revisão de conceitos tradicionais e a elaboração de novos que permitam lidar com a complexidade decorrente dessa convivência entre duas espécies inteligentes, com a perspectiva de num futuro relativamente breve investir a dominância de uma sobre a outra (IA sobre os humanos)[43].

Para Kaufman [43], dentro das questões que são consequência dos recentes avanços, a autonomia dos sistemas inteligentes se caracteriza como uma das mais relevantes, e não em vão uma das mais difíceis pela sua própria natureza: autonomia é uma prerrogativa inerente ao processo de aprendizado das máquinas (*Deep Learning*).

Dois dos conceitos que são referidos na literatura são: responsabilidade social e ética. A literatura apresenta diversos pontos de vista sobre estes conceitos. Para Fischer [46], responsabilidade social e a ética são conceitos muitas das vezes utilizados de maneira indiferenciada. Por exemplo, a responsabilidade social é posta como sendo a ética em um determinado contexto organizacional; a responsabilidade social atua no impacto que a atividade do negócio tem em toda a sociedade, enquanto que o conceito de ética está relacionado com a conduta daqueles que fazem parte da organização.

A literatura não apresenta relação formal entre a responsabilidade social e a ética, apresentando ainda que a ideia de responsabilidade social tem diversas dimensões, entre elas a ética [46]. Um terceiro conceito revisto na literatura é a moralidade. De acordo com Beauchamp e Bowie [47], a moralidade refere-se a princípios ou regras de conduta moral tal como estão definidas na sociedade.

A moralidade existe antes da aceitação ou rejeição dos seus padrões pelos indivíduos, nesse sentido a moralidade não pode ser meramente uma política ou um código. Nesse sentido, a ética tem a ver com o refletir sobre a natureza, e justificações do certo e do errado [47].

Os sistemas baseados em IA, de acordo com Guizzardi et al. [48], não são agentes morais que podem agir com uma razão com base em princípios éticos por si. Pelo contrário, são softwares que tem suas funcionalidades e qualidades baseadas em requerimentos éticos que precisam ser nutridos com base lógica, em adição a outros tipos de requerimentos que são requisitados para o preenchimento.

Existe na literatura uma profusão de artigos que investigam como os sistemas baseados em IA devem ser agentes morais que tomam decisões éticas em tempo de execução [48], por exemplo um carro decidir entre atropelar uma ou mais pessoas, um software não desligar a luz por situação social ou soltar poluentes de um navio não-tripulado em uma área de preservação que mate corais ou peixes em alto-mar. Em contrapartida, estamos interessados em investigar como elicitar requisitos éticos para sistemas baseados em IA para que eles estejam em conformidade e atuem com diretrizes e princípios éticos e outras funcionalidades que devam ter [48].

2.4.1 Princípios Éticos

Múltiplos princípios éticos foram apresentados por diversas instituições públicas, privadas, não governamentais e instituições de ensino. Nestes documentos, os termos princípios e diretrizes às vezes aparecem de formas intercambiáveis. Ao longo deste trabalho assumimos que as diretrizes de ética em IA são um conjunto de princípios éticos. Ou seja, as diretrizes de ética em IA possuem princípios. As principais diretrizes de ética em IA, as mais reconhecidas pela comunidade, são AI HLEG[49] e IEEE EADv1 [50].

Diversos estudos tiveram como objetivo a compilação de princípios éticos presentes na literatura em um único conjunto [51], [19], [52], [53], [54]. Um desses estudos, o qual nos auxiliou a direcionar tais princípios para a implantação de ética em IA, foi elaborado por Ryan e Stahl [7]. Neste trabalho, os autores buscaram por princípios éticos que pudessem auxiliar desenvolvedores e usuários na implementação e entendimento de ética em IA. Esta compilação, feita em 11 princípios é apresentada na Tabela 2.3.

Tabela 2.3: Princípios e suas questões éticas apresentadas no trabalho de Ryan e Stahl [7]

#	Princípio	Questões éticas
1	Transparência	Explainabilidade; Explicabilidade; Compreensibilidade; Interpretabilidade; Comunicação; Divulgação; Apresentação
2	Justiça e equidade (*fairness*)	Consistência; inclusão; igualdade; equidade; não-viés; não-discriminação; diversidade; pluralidade; acessibilidade; reversibilidade; remediar; reparação; contestação; acesso; distribuição
3	Não-maleficência	Segurança; proteção / *safety*; dano; precaução; prevenção; integridade; não-subversão

4	Responsabilidade	Responsabilização / *accountability*; responsabilidade / *liability*; agir com integridade
5	Privacidade	Informação pessoal ou privada
6	Beneficência	Bem-estar; paz; bem social; bem comum
7	Liberdade e autonomia	Consentimento; escolha; auto-determinação; liberdade / *liberty*; empoderamento
8	Confiança	Confiabilidade / *Trustworthiness*
9	Sustentabilidade	Meio ambiente (natureza); energia; recursos (energia)
10	Dignidade	Dignidade
11	Solidariedade	Segurança social; coesão

Neste trabalho iremos utilizar como prova de conceito os princípios baseados no sistema ECCOLA [1], o qual também tem como fundamentações as práticas pregadas pelo IEEE EADv1 [50] e pelo AI HLEG[49].

2.5 Trabalhos relacionados

Foi encontrado na literatura um conjunto de ferramentas que auxiliam a implementação de ética em IA [17]. Neste estudo foi explorado um repositório de código aberto, GitHub, onde foram encontradas 21 ferramentas, disponíveis publicamente que auxiliam a implementação de ética em IA, porem apenas após o sistema ter sido desenvolvido, e com grande foco no principio de Explicabilidade.

Outro estudo [9] encontrou 106 ferramentas, estas porém, em geral, necessitam de adaptação para o contexto do desenvolvedor, ou seja, não são software de prateleira (*off-the-shelf*), focam em partes especificas do processo de desenvolvimento de software. Tais ferramentas não possuem documentação suficiente e usabilidade, além necessitarem de uma forte alteração para se adequarem ao ambiente nos quais deverão ser usadas.

A proposta deste trabalho é semelhante a proposta de Vakkuri et al. [1], no qual um baralho de cartas é utilizado para auxiliar times de desenvolvimento a elicitar requisitos éticos em IA no contexto de desenvolvimento ágil. Assim, nossa proposta envolve a implementação do método ECCOLA, como um Protótipo.

2.6 Método ECCOLA

O ECCOLA consiste em um método que dispõe de 21 cartas no total. Estas cartas são divididos em 8 temas, onde cada tema consiste de 1 a 6 cartas. Estes temas são temas relacionados à ética em IA, encontrados em guias como os tópicos de transparência, dados, agência e supervisão, segurança e prevenção, equidade, bem-estar e responsabilização.

Cada carta, individualmente, aborda os temas com focos mais determinados, como, por exemplo, na área de transparência, como a comunicação está sendo repassada, ou na área de dados, como a privacidade dos dados e a qualidade dos dados são garantidos. Como objetivo, o ECCOLA possui três principais objetivos para a metodologia:

- Ajudar a criar conhecimento sobre a ética em IA e sua importância;
- Criar um método adaptável e modular que se encaixa em diversos contextos de Engenharia de Software, e;
- Fazer o método ser viável para equipes ágeis, e também para fazer a ética ser parte do desenvolvimento ágil em geral.

Primeiramente, são sete princípios presentes no método ECCOLA – baseados primariamente no AI HLEG [49] e no IEEE EADv1 [50] – conforme apresentado na Tabela 2.4, e no protótipo desenvolvido.

Tabela 2.4: Princípios éticos presentes no trabalho de Vakkuri et al. [1]

#	Princípio
1	Transparência
2	Dados (*Data*)
3	Agência e Supervisão (*Agency & Oversight*)
4	*Safety and Security*
5	Equidade (*Fairness*)
6	Bem-estar (*Wellbeing*)
7	Responsabilização (*Accountability*)

O uso dos *cards* se faz durante a fase de estimativa dos projetos na *Sprint*, ajudando assim ao time escolher em quais cartas, ou temas das cartas, que são relevantes para aquela iteração da *sprint*, como se fosse um *Planning Poker*, de acordo com Vakkuri et. al [1]. A depender por quem o ECCOLA é utilizado, ele pode produzir objetivos distintos. Para os *Product Owners*, a ferramenta tende a resultar em histórias de usuário não funcionais envolvendo ética. Para o time de desenvolvedores, o ECCOLA facilita a comunicação com o tema. Usando estas cartas, o time acaba tendo discussões sobre questões éticas e acabam criando decisões baseadas nestas discussões. Finalmente, se utilizado por apenas um desenvolvedor, o objetivo do método é aumentar o conhecimento e atenção em cima de questões éticas na IA.

2.7 Síntese do Capítulo

Vimos neste Capítulo a condensação teórica dos tópicos os quais abordaremos na construção do guia de ética para aplicações no contexto de IA. Verificamos desde as origens da Engenharia de Software, como ela gerou a necessidade de formalizações para as definições em um projeto para a criação de um software, o qual resultou na engenharia de requisitos, área que auxilia a entender e levantar os requisitos necessários para o software, atendendo aos desejos de seu solicitante. Com isso, vimos como o processo de elicitação de requisitos se forma em uma importante área para a correta avaliação de acordo seu *stakeholder*.

Foi verificado também como o desenvolvimento ágil de software nasceu visando que tais softwares não fossem criados apenas de forma rápida, porém também otimizando os recursos de tempo e dinheiro gastos durante a sua criação. Vimos como o Scrum é um dos métodos mais populares, no qual em determinada fase do uso desta metodologia (a fase de planejamento da Sprint), temos o uso de um jogo de cartas que, de forma divertida e de fácil entendimento, ajuda os desenvolvedores avaliar o nível de complexidade de cada história a ser trabalhada em cima. Foram vistos também os conceitos de inteligência artificial e quais são as suas implicações éticas, áreas nas quais o debate é essencial durante a fase de planejamento. No Capítulo 3, será apresentado o desenvolvimento de um guia em forma de *Planning Poker* para auxiliar no debate de quais questões éticas em que a IA a ser desenvolvida deverá se fundamentar, atendendo ao escopo e às necessidades de seus *stakeholders*.

Capítulo 3

Desenvolvimento da Ferramenta

Neste Capítulo será apresentada a segunda fase do *Design Science Research* – Sugestão – onde serão exploradas: a proposta do guia; como ele será trabalhado; como foi realizado o desenvolvimento da ferramenta; como é a interface que será apresentada ao usuário; a forma de se utilizar os *cards* no sistema; e como eles podem ser editados para adequações futuras em outras propostas de ferramentas para elicitação de requisitos éticos em IA.

3.1 Proposta

A ferramenta proposta neste trabalho servirá como um assistente para os *Product Owners* e desenvolvedores elicitarem requisitos éticos em sistemas baseados em IA durante a primeira fase do ciclo de desenvolvimento de software (i.e., a análise de requisitos), no contexto de desenvolvimento ágil de software. Para esta fase de discussão, será utilizado um formato de *Planning Poker* digital, com acesso disponível no repositório no GitHub, em `https://oggvaldo.GitHub.io/eccola`.

3.2 Desenvolvimento

Para o desenvolvimento deste sistema, foram utilizadas as tecnologias de *Hypertext Markup Language* (HTML) [55], *Cascading Style Sheets* (CSS) [56] e JavaScript (JS) [57], com o código devidamente referenciado no apêndice B. Para a apresentação da página inicial e a página que vai apresentar os *cards*, temos o HTML como a base, o CSS como a parte de estilização da página, além de fornecer padrões adaptativos para diversos tamanhos de tela, manutenabilidade e escalabilidade visual do programa sem a necessidade de ferramentas e *frameworks* adicionais, e o JS como a parte responsável pela lógica e ação de funcionalidades do sistema. Com este conjunto de ações, teremos aqui o que será um tipo

de aplicação web, que funcionará diretamente do navegador de qualquer dispositivo com capacidade computacional e que seja compatível com HTML 5 [58].

Com isso, temos um sistema totalmente responsivo a diversos tamanhos de tela. Isto permite seu uso em dispositivos como *smartphones*, *tablets*, *notebooks* e computadores de mesa. Temos esta possibilidade devido ao emprego de *Media Queries* do CSS empregado, que viabiliza a realização da apresentação do conteúdo de forma adaptativa a cada um dos dispositivos supracitados, sem a necessidade de se realizar alterações específicas para cada um dos tipos de equipamentos e/ou telas.

Durante a criação deste guia, alguns cuidados foram tomados. O HTML foi criado de forma a permitir a legibilidade e compreensibilidade com o uso devido do HTML (por exemplo, como as tags semânticas identificam o conteúdo que se encontra nos arquivos do sistema), permitindo assim a leitura do código de forma mais sucinta e de mais fácil identificação. Além disto, o sistema foi projetado também de forma a ser utilizado por pessoas com deficiência visual devido ao emprego de tal técnica. Para tanto, este sistema está satisfatoriamente adaptado ao uso de sistemas ledores de tela, como o *Non Visual Desktop Access* (NVDA).

Simultaneamente com a disponibilização do site estático com o uso do HTML e do CSS, o JS foi utilizado para trazer ao sistema os elementos que fornecem a interatividade e o dinamismo do programa para os usuários.

Assim, foi possível dar ao usuário as possibilidades de seleção e comparação dos *cards*, filtragem de acordo o princípio ético que pretendem explorar e a facilidade na realização de alterações do sistema para outros tipos de elicitações a serem inseridos na estrutura da aplicação, permitindo, desta maneira, a futuros desenvolvedores a possibilidade de modificar e adicionar *cards* e princípios éticos com baixa complexidade, objetivando um sistema com a qualidade de *user-friendly*, tanto para quem irá operar (e.g., Product Owners, desenvolvedores de sistemas baseados em IA) quanto para quem irá editar o sistema (e.g., pesquisadores de ética em IA) através do acesso aberto ao código do sistema localizado no repositório, onde o fonte se encontra disponível.

Para o caso de um projeto para servir como template, para futuras alterações, o mesmo se encontra disponível em `https://www.GitHub.com/oggvaldo/aiethicsguide`, contendo mais informações e exemplos de como se alterar o guia conforme o usuário desejar.

Com isso, temos em mente o objetivo de contemplar os princípios éticos aplicáveis à área de ética em IA com o fornecimento do sistema e de seu código-fonte, compreensibilidade e instruções de uso, explorando o princípio ético da transparência, com foco nas questões éticas de divulgação, comunicação, apresentação, explicabilidade, compreensibilidade e interpretabilidade, e, ao adequar a tecnologia para o uso de *softwares* leitores de

tela, temos a contemplação do princípio de beneficência e dignidade atendidos segundo os princípios éticos definidos por Ryan e Stahl [7].

Com todos os aspectos técnicos de software descritos anteriormente, na entrada do sistema o usuário poderá ler como utilizar os *cards* em um contexto de desenvolvimento ágil com o uso da técnica de *Planning Poker*, conforme a figuras abaixo apresenta.

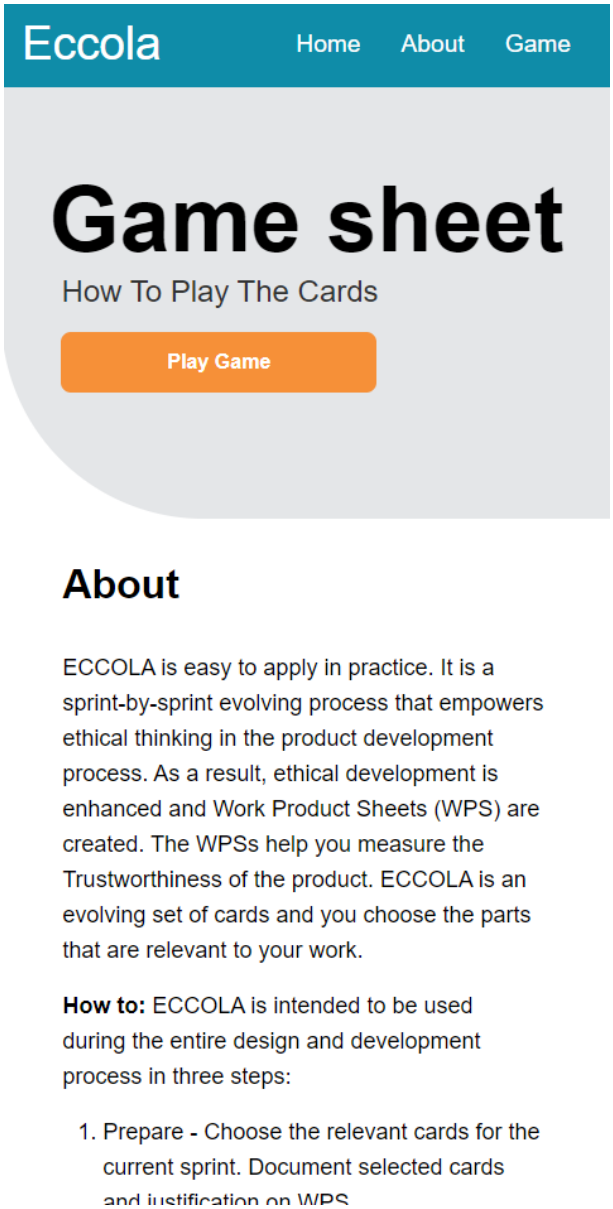

Figura 3.1: Imagem do sistema adaptado para telas de *smartphones* e *tablets*

Figura 3.2: Imagem do sistema adaptado para telas de computadores de mesa e *notebooks*.

3.2.1 Interface

Para a criação deste guia, a partir da inspiração de um *Planning Poker* voltado para o debate de projetos, foi concebido que o sistema deveria apresentar seus tópicos através do formato de *cards*. Vendo como o ECCOLA [1] fez a implantação em forma analógica (*cards* físicos) do modelo e o mesmo pode ser aplicado para qualquer outro conjunto de regras de elicitação de requisitos de ética em IA, foi julgado que adotar o modelo de *cards* para a criação deste guia seria ainda o ideal, que também possuem imagens em mais detalhes no apêndice A.

Para tanto, usamos a plataforma de desenvolvimento de protótipos FIGMA, na qual foi possível desenhar como seria o formato do guia e, a partir dele, criar um *mockup* do que viria a ser o guia. Após a realização de um *brainstorm* sobre como desenvolver o projeto, foi entendido que era necessário que o sistema apresentasse ao menos três telas, sendo uma de boas vindas do sistema, apresentando as regras do jogo, o que é e o título do sistema, uma com os *cards* a serem mostrados aos participantes do planejamento, e uma tela onde se apresentam os *cards* selecionadas apenas, com a opção de retorno e selecionar novamente os *cards*. Ao término do projeto de prototipação, chegamos no desenho atual do guia, que pode ser acessado clicando aqui.

Com fortes inspirações nos modelos utilizados pelo design gráfico criado pelo Google, o Material Design [59] influenciou fortemente na construção da interface dos *cards*, ajudando-nos a adotar um modelo de *card* que pudesse ser facilmente replicado e identificado, sem perder a característica de um visual limpo, informativo e agradável visualmente,

independente da plataforma ao qual o guia esteja sendo utilizado.

Figura 3.3: Imagem do protótipo do sistema na tela inicial criado na ferramenta Figma.

Para a tela de boas vindas (figura 3.3), um design com poucas distrações, uma descrição de como utilizar o sistema e a possibilidade de se inserir um arquivo mostrando como trabalhar com os *cards* (uma vez que estes são pensados para serem utilizadas em projetos variados de elicitação de requisitos) ou outras descrições. Para tanto, basta que um usuário com conhecimentos em HTML altere a página inicial ao seu gosto, de acordo as necessidades criadas pelo projeto a ser utilizado junto ao guia.

Figura 3.4: Imagem do protótipo do sistema na tela de seleção de *cards* criado na ferramenta Figma.

Para a tela de escolha dos *cards* (figura 3.4), temos todos os *cards* apresentados, com um menu *drop-down* para o filtro de seleção da categoria dos *cards* a ser escolhida para a rodada de debate. Estes *cards* e suas categorizações serão melhor detalhadas na seção 3.3. Temos também o botão *Compare cards*, onde, após a seleção dos itens para o debate, teremos o prosseguimento para a tela de resultados após apertar o botão.

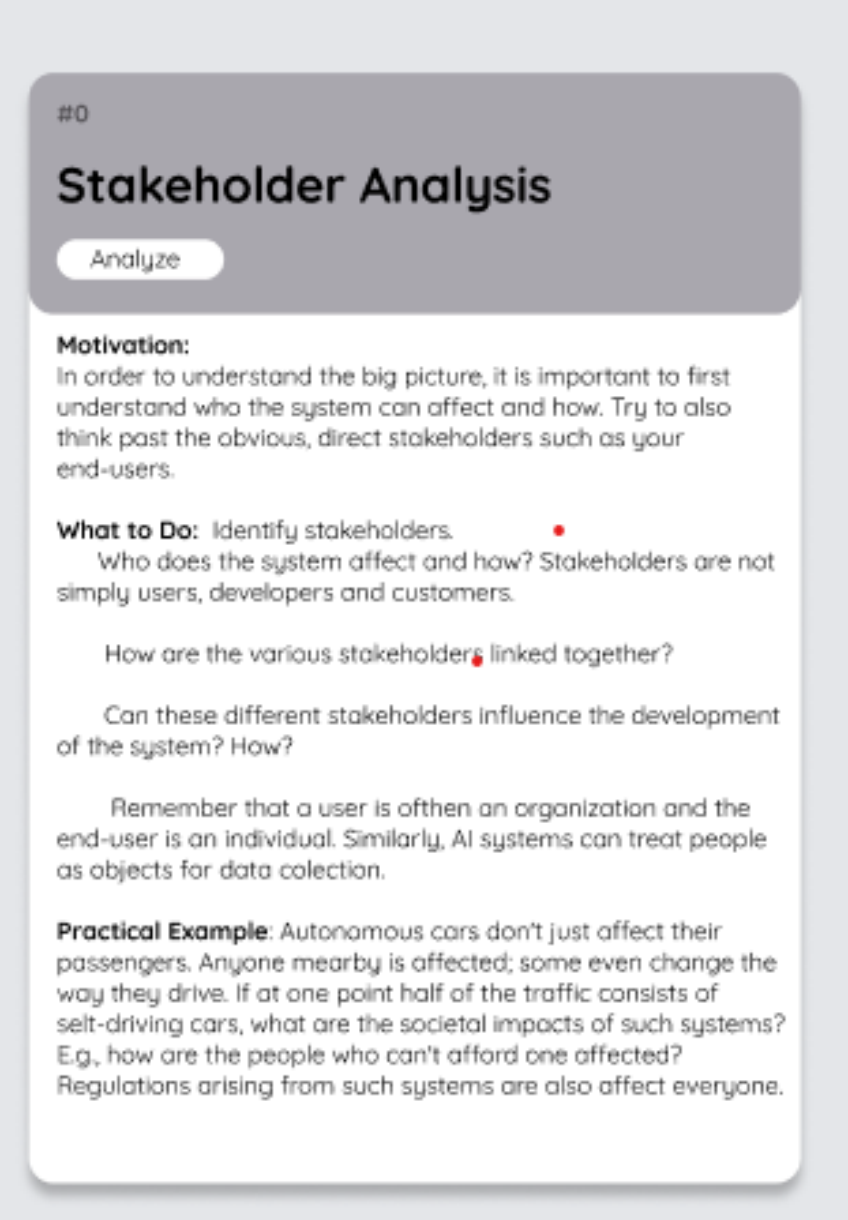

Figura 3.5: Imagem do protótipo de um dos *cards* em detalhe criado na ferramenta Figma.

Para a tela de resultados (figura 3.6), durante a prototipação, foi deixada propositalmente em branco, porém na versão atual do sistema temos a apresentação dos *cards* selecionados na tela anterior juntamente com o botão *Start again*, que permite, caso queira, realizar uma nova seleção dos itens para o debate, o que será abordado em maiores detalhes na Seção 3.3.

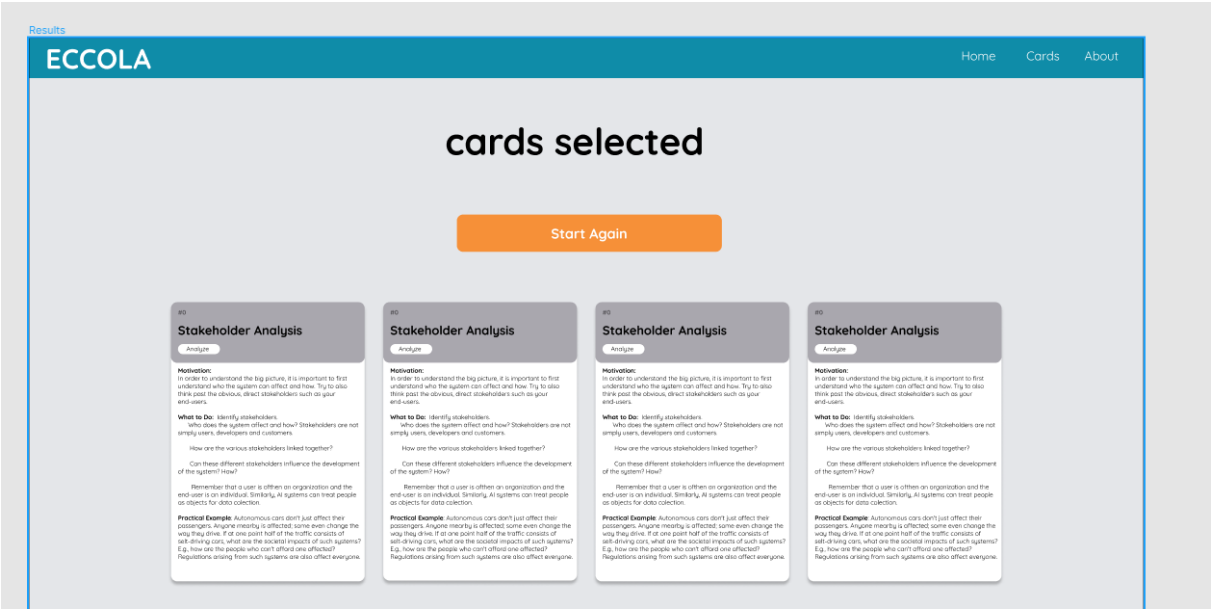

Figura 3.6: Imagem do sistema na tela de *cards* selecionados.

3.3 Como utilizar

Tomando como referência o ECCOLA, apresentado na figura 3.2, o sistema terá a tela inicial apresentando um botão "Play game", seja no topo da tela ou ao final (inclusive no modo para dispositivos móveis), e os links Home, About e Game no topo da tela, onde cada um dos links envia para uma das páginas citadas (as quais podem ser customizadas e serão detalhadas em maior profundidade na Seção 3.5). Neste caso, o link *Home* encaminha para a página inicial, o link *About* para o espaço onde está a descrição do jogo a ser inserida e o link *Game* para a página onde se encontram os *cards*. Os botões *Play the Game*, localizados tanto no início quanto no final da página, tem o mesmo propósito do link *Game*, levando também para a página onde o *Planning Poker* será realizado.

Na interface do *Planning Poker*, como mostrado na figura 3.7, temos um menu *dropdown* que serve como filtro, onde somente os *cards* do tipo selecionado estarão sendo exibidas na tela (caso o filtro *ALL* esteja ativo, todos os *cards* estarão exibidos), o botão *Compare Cards*, que será acionado somente em caso de selecionar dois *cards* para a

discussão (onde o número de *cards* também pode ser alterado, como veremos adiante) e logo abaixo os *cards* que estarão disponíveis para a seleção e discussão. Ao selecionar em cada um dos *cards*, eles terão uma borda laranja demonstrando a seleção dos mesmos, e após selecionar no botão *Compare Cards* teremos a tela mostrando apenas os *cards* selecionados. Estes serão os *cards* a serem utilizados no debate, e será apresentado como utilizá-los na seção 3.3.1.

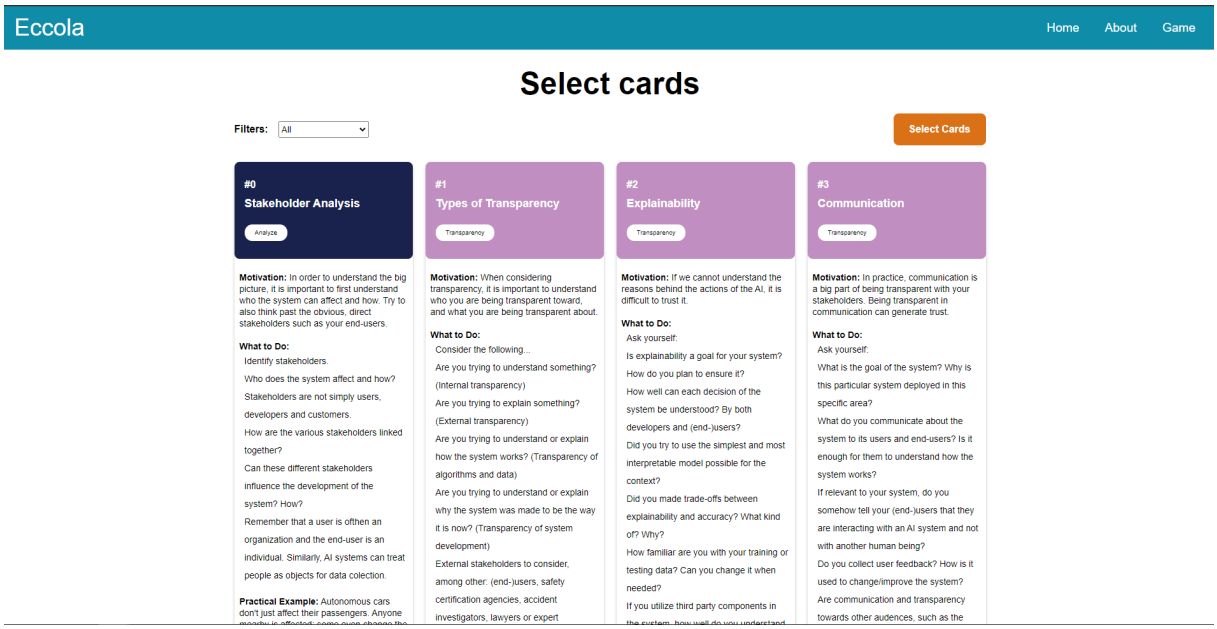

Figura 3.7: Imagem do sistema com os *cards* disponíveis em uma tela com resolução maior que 640 pixels.

3.3.1 Utilizando os *cards*

Para a utilização do sistema, é necessário retomarmos aos conceitos de um *Planning Poker* e adaptarmos ao uso no contexto de ética em IA. Para um melhor entendimento, temos que também adaptar o debate ao ambiente de desenvolvimento ágil e o uso do Scrum, já que sem estas ferramentas o uso deste guia fica muito mais limitado.

Para tanto, repensando o contexto de planejamento de um projeto, temos que ter, desde o princípio dele, o entendimento de que o mesmo funcionará através de iterações contínuas e constantes, as quais devem sempre estar contextualizadas durante o progresso de seu serviço, porém também mantendo a liberdade do usuário adaptar a ferramenta de acordo o contexto no qual se encontra no momento de uso.

Desta forma, este guia tem como função balizar os desenvolvedores e *stakeholders* a terem um melhor entendimento de como aplicar os princípios éticos da IA ao projeto que

estiver sendo implantado durante a *sprint*, uma vez que, apesar de haver a implantação técnica de um lado, através de como será desenvolvido o algoritmo e em qual linguagem, temos também na abordagem do assunto ético diretrizes que vão guiar o comportamento futuro da aplicação perante o ambiente no qual esta estará inserida e em pleno funcionamento.

Portanto, acima de tudo, o uso dos *cards* vai servir para que os desenvolvedores possam entender, avaliar, mensurar e revisar a extensão que sua aplicação possa vir a apresentar durante seu ciclo de vida, permitindo assim que este guia seja uma ferramenta que se encaixa plenamente no conceito de desenvolvimento ágil junto com as ferramentas pré-existentes.

No sistema, teremos os *cards* dispostos de acordo com o seu proponente apresentar para a equipe. Para fins de conceito, estaremos utilizando os *cards* descritos no sistema ECCOLA [1], uma vez que estes se baseiam nos princípios citados por Ryan e Stahl [7].

Para o uso dos *cards* em si, independente das regras pré-definidas, é necessário que sempre haja pelo menos dois *cards* escolhidos para o debate. Verifica-se esta necessidade por, segundo Ryan e Stahl [7], termos onze princípios que balizam, praticamente, todos os pontos de debate que veremos em qualquer outro guia a ser montado com auxílio desta ferramenta. E, não obstante o enriquecimento da discussão durante o desenvolvimento da parte técnica da ferramenta pode ser aprimorado com o uso adequado de mais de um *card*, uma vez que haverá pontos de discussão que irão cruzar entre si e delimitar em um ponto que ajudará a indicar se o desenvolvimento da ferramenta em debate está adequado para o contexto em que se encontra, podendo assim criar uma folha de trabalho do produto.

Uma vez selecionados os *cards*, e aberta a discussão, faz-se a revisão dos princípios de cada *card* e princípio contido no *card*, visando, assim como no modelo de desenvolvimento Scrum e o andamento de uma *sprint* (preferencialmente durante a discussão dos épicos e histórias), sempre preparar o ambiente com uma visão inicial e um apontamento pré-definido, a revisão do caminho tomado, já que ajustes e correções são necessárias ao longo da implantação do projeto, assim como na parte de implantação do código, para melhor chegar na fase final de avaliação, e verificar se os pontos foram devidamente atendidos.

Durante cada um destes processos, escrever a parte como foi o pensamento desenvolvido em cima do princípio ético, o qual ajudará na fase de retrospectiva da *sprint*, servindo de roteiro do que foi trabalho, do que não foi e servindo de aprimoramento para a próxima realização.

Com isso, podemos resumir os pontos debatidos acima em três:

1. Preparar: Escolher os *cards* para a *sprint* corrente, documentar e justificar suas escolhas em uma folha de trabalho do produto.

2. Revisar: Manter os *cards* selecionados à vista durante as tarefas e escrever se alguma ação foi tomada baseada nos *cards*.

3. Avaliar: Revisar o trabalho feito para garantir que as ações planejadas foram tomadas. Revisar os *cards*, e, se necessário, revisar as tarefas.

Como observado por Vakkuri et al. [1], é recomendado que a cada *sprint* estes processos sejam repetidos em todas as iterações, e chegando ao final da *sprint* que seja realizada uma retrospectiva, discutindo o que foi trabalhado, o que não foi trabalhado e quais as partes relevantes para a próxima *sprint* do desenvolvimento do produto.

3.4 Documentação

Temos nesta seção a descrição do sistema feito como um *webapp*, mostrando trechos do código e como ele se encontra em cada um dos arquivos. De uma forma geral, temos no arquivo **index.html** a página inicial do guia, contendo a tela inicial e uma pequena instrução de como utilizar o sistema, o arquivo **game.html** contendo a página que receberá os *cards* do sistema, os arquivos .css *game*, que apresentam a estilização dos *cards*, *style*, responsável pelos aspectos gráficos dos HTML, e *reset*, a função que ajuda a jogar novamente o *Planning Poker* sem necessitar recarregar a página para tal. Estaremos a seguir descrevendo cada um dos arquivos com maiores detalhes, demonstrando alguns de seus detalhes técnicos na produção deste sistema e como cada módulo foi pensado para a criação, manutenção e atualização futura.

Para os usuários que desejam editar o sistema de forma mais personalizada, abaixo seguiremos com a documentação do código, o que permitirá, aos que assim desejarem e possuírem um conhecimento intermediário na leitura de código HTML, CSS e JavaScript, uma melhor ideia em quais trechos do código editar.

3.4.1 *index.html*

Este é o arquivo responsável por apresentar a tela inicial do sistema. Este módulo do sistema é a parte que mostra as informações iniciais do sistema, como a tela de apresentação, um pequeno *how-to* e a tela para prosseguimento ao *Planning Poker*. Abaixo iremos verificar trechos que permitem as alterações conforme os usuários assim desejarem, apresentando por cima o que cada parte deste código faz.

Temos o código responsável pelo cabeçalho da página, o qual também é aproveitado na parte do *game.html*. Esta parte facilita àqueles que desejam inserir mais páginas para navegação, como uma parte de “sobre o sistema” ou alguma outra página auxiliar.

Listing 3.1: Código do nav.

```
1  <nav class="nav">
2      <a  href="index.html" class="nav__logo">Eccola</a>
3      <ul class="menu">
4        <li class="menu__item"><a href="index.html">Home</a>
           </li>
5        <li class="menu__item"><a href="#about">About</a></
           li>
6        <li class="menu__item"><a href="game.html">Game</a><
           /li>
7      </ul>
8  </nav>
```

A seguir, temos a parte responsável pela página inicial com a arte do sistema, apresentando o título, um nome auxiliar que pode ser livremente alterado (foi deixado *How To Play The Cards* para quesito de chamar atenção), desde os textos até a imagem. É recomendado realizar poucas alterações nesta parte, visto ter sido pensada para chamar atenção para o uso dos *cards*, de forma a apresentar uma tarefa em formato mais divertido e lúdico. Temos aqui também o botão *Play Game*, que encaminha para a tela com os *cards* apresentados.

Listing 3.2: Seção principal

```
1
2    <section class="principal">
3      <div class="content__title">
4        <h1 class="title">Game sheet</h1>
5        <h3 class="subtitle">How To Play The Cards</h3>
6        <a href="game.html" class="button__game">Play Game</
           a>
7      </div>
8      <div>
9        <picture>
10         <img
11           class="illustration"
12           src="./assets/img/card-game.svg"
13           alt="Imagem de uma mao segurando tres cartas,
                uma na cor azul escuro, laranja e azul claro"
14         />
15       </picture>
16     </div>
```

```
17        </section>
```

As duas sequências de código acima complementam a parte de *header* do sistema.

Em seguida, temos a parte da seção *About*, onde são apresentadas as instruções de como usar o guia, e em seguida mais um botão *Play Game*, encaminhando para a tela de seleção de *cards*. Este botão foi inserido visando atender os usuários que se encontram em plataforma *mobile*, não precisando assim rolar toda a tela para o início da página novamente para iniciar o jogo.

Listing 3.3: Código apresentando o *about* e como usar o guia.

```
1  <main class="container main">
2        <h2 class="about__title" id="about">About</h2>
3        <p class="about__paragraph">
4          About ECCOLA
5        </p>
6        <div class="explication_container">
7          <div class="explication">
8            <p class="about__paragraph">
9              <span class="strong">How to:</span> ECCOLA is
                  intended to be used during the entire design
                  and
10             development process in three steps:
11           </p>
12           <ol start="1" class="about__list">
13             <li>
14                 Example
15             </li>
16           </ol>
17         </div>
18         <div class="explication_img">
19           <picture>
20             <img src="./assets/img/about.jpg" alt="" srcset=""
                   />
21           </picture>
22         </div>
23         <p class="about__paragraph">
24           <span class="strong">Practical Tip: </span> Repeat
                the process in every iteration. Remember to do
25           a retrospective afterwards. Think about what worked
                & what did not.
```

```
26          Choose the parts that are the most relevant for your
                workin the next
27          round.
28        </p>
29      </div>
30    </main>
```

Para fins de praticidade, foi removida a parte do texto explicando o que é o ECCOLA e como usar o mesmo. Com isso concluímos a apresentação da tela inicial do sistema.

3.4.2 *game.html*

Aqui temos a tela que vai receber o motor do sistema, este desenvolvido em JavaScript e a ser abordado em detalhes na parte 3.4.4. Temos, mais uma vez no começo do código, a referência da barra de navegação com as mesmas características apresentadas na página inicial, porém a partir do corpo do código temos fortes mudanças, como a parte de filtragem dos *cards*, botão para a confirmação da seleção dos *cards* e a mensagem para o número mínimo de seleção de *cards*.

Listing 3.4: Seleção dos *cards*.

```
1       <div class="container__game container">
2         <h1 id="game__title">Select cards</h1>
3       </div>
```

Aqui o código que simplesmente mostra o botão para a seleção dos *cards* escolhidos pelo usuário.

Listing 3.5: Filtro dos *cards*.

```
1         <div class="container__filters">
2           <label for="filterCards" class="filter__paragraph">
                Filters:</label>
3           <select name="filterCards" id="filterCards">
4             <option value="all">All</option>
5             <option value="Analyze">Analyze</option>
6             <option value="Transparency">Transparency</option>
7             <option value="Safety & Security">Safety &
                Security</option>
8             <option value="Fairness">Fairness</option>
9             <option value="Data">Data</option>
10            <option value="Agency & Oversight">Agency &
                Oversight</option>
```

```
11              <option value="Wellbeing">Wellbeing</option>
12              <option value="Accountability">Accountability</
                    option>
13            </select>
14          </div>
```

Neste *snippet*, apresentamos a parte de filtragem dos *cards* que serão apresentados no sistema. Este é um dos trechos que se faz necessário editar caso o usuário queira inserir novos tipos (ou valores) de *cards*, o que será explicado em detalhes na seção 3.5.

Listing 3.6: Alerta de seleção e área de criação dos *cards*.

```
1     <section class="container">
2         <div class="container__alert">
3           <p class="alert">select at least two cards</p>
4         </div>
5         <div class="cards"></div>
6     </section>
```

Trecho onde o sistema apresenta o alerta de que se o número mínimo de *cards* não foi selecionado não haverá o prosseguimento para a tela com os *cards* selecionadas. Adicionalmente, temos a classe responsável por apresentar os *cards* na tela também.

Listing 3.7: Motores para o funcionamento do sistema.

```
1     <script src="./assets/js/infosCards.js"></script>
2     <script src="./assets/js/index.js"></script>
```

Motores do sistema, que serão explicados em maiores detalhes adiante.

3.4.3 *game*, *style* e *reset.css*

Neste tópico, passamos pelo arquivo que apresenta a estilização do sistema. Uma vez que várias partes do código são elementares, estaremos apresentando os principais trechos para a alteração futura do sistema.

Os arquivos *reset.css* e *style.css* são arquivos que devem ser alterados somente caso queira modificar o comportamento estético do sistema, tendo em vista o *reset.css* servir puramente para limpar a tela e apresentar novamente os *cards* caso o botão *Play Again* seja pressionado. No caso do arquivo *style.css*, ele apresenta somente as características de estética do sistema, tanto para a cor das barras de cabeçalho e rodapé quanto a cor dos botões. Aqui também temos parte do código que permite ao sistema ter um comportamento adaptável a diversos tamanhos de tela, apresentando somente uma carta por vez na tela e os textos nas opções com telas a partir de 360 até 479 *pixels*, dois *cards*

simultâneos na tela e outras características gráficas de 480 até 779 *pixels*, e a partir de 780 *pixels* apresentando tanto a tela inicial completa quanto três ou mais *cards*, conforme o tamanho do monitor disponível para a exibição do sistema.

Listing 3.8: Exemplo de código que apresenta a adaptabilidade do sistema de acordo a pixelagem disponível da tela

```
@media screen and (max-width: 1024px) {
  .title {
    font-size: 72px;
    line-height: 90px;
  }
  .subtitle {
    font-size: 24px;
  }

  .illustration {
    width: 300px;
    height: 300px;
    position: relative;
    bottom: -2px;
  }

  .explication_container {
    flex-direction: column;
  }
  .explication_img img {
    height: 400px;
  }

  .container {
    max-width: 960px;
  }
}
```

No caso do arquivo *game.css*, dispomos aqui o *layout* e a adaptabilidade da quantidade de *cards*, além de sua posição na tela de acordo a pixelagem e tamanho disponível. Porém, aqui nesta tela, a função mais importante se dá para a identificação dos *cards*, que são as indicações do nome da classe ao qual o *card* terá. Nesta função em específico, é necessário que o nome esteja sempre em minúsculo e, caso haja mais de uma palavra, esta classe seja

escrita com hífen no lugar do espaço, devido a uma limitação do CSS na interpretação do espaço para o nome das funções.

Listing 3.9: Exemplo de carta do tipo *Agency & Oversight* contendo nome composto.

```
1 .agency-oversight {
2   background-color: #5aaead;
3 }
```

3.4.4 *index.js*

Temos aqui o que é o motor de todo o sistema. Neste arquivo, encontramos os módulos responsáveis por toda a organização, exibição dos *cards* e funcionamento das opções para que o sistema se comporte de forma ideal. Sem a parte de JavaScript não teríamos as utilidades deste webapp, por isto este é considerado o principal motor deste guia, dando a cada um dos botões, *cards* e menu de filtragem as suas devidas utilidades. Temos ao total 18 funções que criam as classes para o correto funcionamento do guia. Estaremos logo a seguir descrevendo o funcionamento de cada uma destas funções e como as classes realizam os trabalhos devidos, porém graças à técnica de código autodocumentável temos um código que praticamente não necessita de comentários nele, atendendo às melhores práticas de programação. Também, para fins didáticos, o código se mostra o mais limpo e com as definições mais claras possíveis, permitindo assim também a colaboração por parte de terceiros que queiram atuar no o projeto futuramente. A seguir temos uma breve documentação deste arquivos para servir de guia a projetos futuros. Para o melhor entendimento destes arquivos, é recomendada a análise por alguém que possua um conhecimento em lógica de programação (para o entendimento lógico de como são as chamadas de função) e de JavaScript para a edição, com espaço para futuras otimizações de código.

Este é o trecho do código responsável por apresentar os aspectos visuais da página, permitindo as seleções dos botões, o funcionamento dos filtros, a exibição de alertas (em caso de escolha de menos *cards* que o requisitado), a chance de selecionar e re-selecionar os *cards*, mostrar apenas os *cards* escolhidos na tela (após selecionar no botão *Compare Cards*) e mostrar apenas os *cards* filtrados. Devido ao código apresentar variáveis autodeclaráveis, o entendimento deste trecho do código se torna elementar.

Listing 3.10: Funções que criam o aspecto visual do guia

```
1     const filter = document.querySelector("#filterCards");
2     const filterContainer = document.querySelector(".
          container__filters");
3     const buttonResult = document.querySelector("#result");
4     const titlePage = document.querySelector("#game__title");
```

```
5       const startAgain = document.querySelector("#start");
6       const container = document.querySelector('.
            container__options');
7       const alert = document.querySelector('.alert')
```

Abaixo, temos a função que realiza a visualização dos *cards* e faz a ativação do botão de resultado ao ser clicado. A função *document.querySelectorAll* é a responsável por fazer a visualização de todos os *cards* que se encontram dentro do arquivo 3.4.5, que será detalhada a seguir. Assim, ao selecionar em um *card*, ele se encontra com uma borda laranja, mostrando que se encontra selecionado e pronto para ir para a tela de resultado.

Listing 3.11: Função que seleciona os *cards* e ativa o botão de resultado

```
1  const selectedCards = () => {
2    const cards = document.querySelectorAll(".card");
3    cards.forEach((card) => {
4      card.addEventListener("click", function () {
5        this.classList.toggle("active");
6      });
7    });
8  };
```

Esta função é a responsável por mostrar os *cards* selecionados em uma nova tela após o botão de *Compare Cards* ser selecionado. Adicionalmente, traz também o botão *Start Again* para que, após o clique, seja possível selecionar novamente os *cards*. Aqui ainda é possível selecionar os *cards* para deixá-los com contorno laranja, mostrando atividade neles, o que pode ajudar na discussão daquele *card* selecionado.

Listing 3.12: Função para mostrar os *cards* selecionadas

```
1  const statePageResult = () => {
2    buttonResult.style.display = "none";
3    filterContainer.style.display = "none";
4
5    startAgain.style.display = "block"
6    container.style.display = "flex";
7    container.style.justifyContent="center";
8
9    titlePage.innerText = "Result";
10   alert.style.display = "none";
11 };
```

Esta é a função responsável para que ao selecionar no *card* ele tenha o contorno laranja, mostrando que o *card* foi devidamente selecionado e está pronto para ir à tela seguinte.

Listing 3.13: Função para selecionar os *cards*

```
const statePageSelect = () => {
  buttonResult.style.display = "block";
  titlePage.innerText = "Select cards";

  startAgain.style.display = "none";

  filterContainer.style.display = "flex";
  container.style.display = "flex";
  container.style.justifyContent="space-between";
  alert.style.display = 'none'
};
```

Esta é a função responsável por transformar o menu *drop-down* em um filtro, permitindo que, após escolher alguma opção do menu contido na seção 3.4.2, somente os *cards* que tenham aquele tipo também sejam mostradas na tela de seleção de *cards*.

Listing 3.14: Função que faz os filtros dos cards

```
filter.addEventListener("click", (event) => {
  if (event.target.value === "all") {
    return createCards(infosCards);
  }

  statePageSelect();

  const cards = filterCards(infosCards, event.target.value);
  return createCards(cards);
});
```

Abaixo, a função para filtrar os *cards* de acordo o seu tipo sendo utilizada.

Listing 3.15: Função que filtra o array

```
const filterCards = (array, type) => {
  return array.filter((infos) => {
    return infos.type === type;
  });
};
```

Função que transforma os *cards* que tenham nomes compostos em nomes simples, devido à limitação do CSS em não reconhecer palavras separadas. Sempre que uma carta com mais de um nome for criada é necessário inserir nesta função o nome dela conforme o padrão para ser reconhecido pelo JavaScript.

Listing 3.16: Função que formata a classes de CSS

```
const formatClasse = (type) => {
  if (type === "Safety & Security") {
    return "safety-security";
  }

  if (type === "Agency & Oversight") {
    return "agency-oversight";
  }

  return type.toLowerCase();
};
```

Função que serve para criar o padrão (*template*) dos *cards* e os listar devidamente.

Listing 3.17: Template de List do *card*

```
const templateList = (list) => {
  return list.reduce((accumulator, actual) => {
    return (accumulator += `
      <li>${actual}</li>
     `);
  }, "");
};

const templateListLink = (list) => {
  return list.reduce((accumulator, actual) => {
    return (accumulator += `
      <li>
      <a href="${actual.link}">${actual.descricao}</a>
      </li>
     `);
  }, "");
};
```

Esta é a função que vai dar o formato de *card* na página ao objeto utilizado anteriormente. É uma função que utilizará boa parte das funções criadas anteriormente para dar a ordenação e orientação ao código 3.4.2.

Listing 3.18: Função que pega template criado e adiciona no DOM

```
const insertCardsIntoPage = (template) => {
  const cards = document.querySelector(".cards");
```

```
3     cards.innerHTML = template;
4     // Adiciona funcionalidade de selecionar os cards
5     selectedCards();
6   };
```

É a função que permite, de maneira mais prática e dinâmica, a inserção dos *cards* no guia. Aqui temos as informações a serem lidas pelo código *infoCards.js* (3.4.5). A partir desta função o usuário tem a liberdade de adicionar mais ou menos funções para os futuros *cards* que serão utilizados pelo guia. Esta função pode ser interpretada como a função que vai mostrar o conteúdo do *card* de maneira parametrizada e mantendo a uniformidade do tamanho dos *cards* independente da quantidade de informação que tenha dentro dele. Como falado anteriormente, aqui inserimos as características que tem nos *cards* do EC-COLA, porém, visando praticidade para a consulta em busca de um material adicional, inserimos no código o parâmetro *link*, onde é possível inserir um link de referência ao *card* e seu devido redirecionamento fica centralizado no código *infoCards.js* (3.4.5). É possível verificar que todos os *cards* tem a possibilidade de apresentar campos em branco, uma vez que nem todos os *cards* apresentam todas as informações de forma padrão, dando assim ao usuário do guia liberdade também para inserir ou retirar informações de cada *card*, a depender de sua estruturação teórica.

Listing 3.19: Função para a criação do conteúdo de cada um dos *cards*

```
1  const createCards = (infos) => {
2    const template = infos.reduce((accumulator, actual) => {
3      return (accumulator += `
4      <article class="card">
5      <div class="card__header ${formatClasse(actual.type)}">
6        <p class="card__number">#${actual.number}</p>
7        <h3 class="card__title">${actual.title}</h3>
8        <span class="card__badge">${actual.type}</span>
9      </div>
10     <div class="card__body">
11       <p>
12         <span class="strong">Motivation: </span>${actual.
               motivation}
13       </p>
14       <br>
15       <ul>
16         <span class="strong">What to Do:</span>
17         <br>
18         ${templateList(actual.whatToDo)}
```

```
      </ul>
      <br>
      <p>
        ${
          actual.praticalExample
            ? `<span class="strong"> Practical Example:</span>
                ${actual.praticalExample} `
            : ""
        }
      </p>
      <p>
      ${
        actual.links
          ? `<p class="strong"> Ferramentas:</p>
             <ul>
             ${templateListLink(actual.links)}
             </ul>
            `
          : ""
      }
    </p>

    </div>
  </article>
    `);
  }, "");
  insertCardsIntoPage(template);
};
```

Esta é a função que mostrará todos os *cards* selecionados anteriormente na página, após o clique no botão *Compare Cards*. Percebe-se que aqui é também onde se altera o número de *cards* mínimos necessários para a não exibição de uma mensagem de erro, informando para selecionar o número de *cards* escolhido pelo usuário (no caso apresentado, dois), e continuar para a tela com os *cards* selecionados previamente.

Listing 3.20: Função que captura todos *cards* que estão ativos e transforma em um array.

```
buttonResult.addEventListener("click", () => {
  const selectedCards = document.querySelectorAll(".active");
  const cards = document.querySelector(".cards");
  if (selectedCards.length >= 2) {
```

```
5       cards.innerHTML = "";
6       selectedCards.forEach((card) => {
7         card.classList.remove("active");
8         cards.appendChild(card);
9       });
10      statePageResult();
11      return
12    }
13    return alert.style.display = 'block'
14
15  });
```

Aqui, a função responsável por selecionar o *card* após o clique.

Listing 3.21: Função para seleção de *card* após clique.

```
1  startAgain.addEventListener("click", () => {
2    statePageSelect();
3    createCards(infosCards);
4  });
```

E por último, a função que realiza a construção dos *cards* a partir da leitura do arquivo *infosCards.js*, logo após iniciando o carregamento da página.

Listing 3.22: Construindo os *cards*.

```
1  const init = () => {
2    createCards(infosCards);
3  };
4
5  init();
```

Como pudemos analisar, nesta sessão temos o código fonte do motor responsável pelo devido funcionamento do guia. A seguir, iremos analisar como se dá a construção dos *cards* no juntamente ao arquivo *infosCards.js*.

3.4.5 *infosCards.js*

Este é, juntamente com o código *index.js* (3.4.4), um dos principais módulos do guia, uma vez que nele vamos inserir as informações necessárias para a construção dos *cards*. Para fins de prova, estaremos dissecando tipo a tipo as variáveis desta classe, permitindo assim a construção de mais *cards*. Vamos utilizar o *card* 20 do ECCOLA, por este conter todos os campos que são utilizados neste guia, adicionando um campo *link* para ser usado em guias

futuros para fins de redirecionamento para mais documentação quanto ao entendimento do *card* em voga.

Listing 3.23: Informações referentes ao *card* 20 do ECCOLA.

```
const infosCards = [
 {
    number: "20",
    type: "Accountability",
    title: "Minimizing Negative Impacts",
    motivation:
      "Minimizing negative impacts of the system is
         financially important for any developer organization.
          Incidents are often costly.",
    whatToDo: [
      "First, consider:",
      "Is your stakeholder analysis is up-to-date {Card #0}",
      "Have you discussed risks? {Card #13}",
      "Have you discussed auditability? {Card #18}",
      "Have you discussed redress issues? {Card #19}",
      "Are the people involved with the development of the
         system also involved with the development of the
         system also involved with it during its operational
         life? If not, they may not feel as accountable.",
      "Are you aware of laws related to the system?",
      "Can users of the system somehow report volnerabilities,
          risks and other issues in the system?",
      "  With whom have you discused accountability and other
         ethical?",
    ],
    praticalExample: "",
    links: [
      {
        descricao: 'Google',
        link: 'http://google.com.br'
      },
      {
        descricao: 'Facebook',
        link: 'http://facebook.com.br'
      }
    ]
```

```
30    },
31
32 ];
```

- Number: Nesta variável inserimos qual o número do *card* simplesmente. Apenas para facilitar a fim de ordenação visual do guia. Campo puramente estético.
- Type: Neste campo, inserimos qual é o tipo que o *card* adotará. Neste campo, é importante que o nome no arquivo referente à coloração do *card* no código *game.css* esteja todo em minúsculo e, caso tenha mais de um nome, os espaços estejam sendo substituídos pelo carácter "-". Lembrando que este nome também deve se encontrar escrito de forma igual no filtro presente no arquivo 3.4.2
- Title: Este campo se dá ao que será discutido com este *card* selecionado.
- Motivation: Campo que se dá a descrição do título ao *card*. Aqui é onde vem a breve descrição do título do *card* que, ao ser selecionado, dará um norte ao que será discutido.
- What to do: Campo em que propõe a forma a se discutir o *card* selecionado. Aqui é onde se descreve por tópicos quais são os pontos que vão dar o norte durante a sessão de discussão em cima do tema escolhido e definido pelo *card*.
- Practical Example: Caso aplicável, um exemplo prático da discussão feita em cima do *card*.
- Links: Parte onde é possível se inserir *hyperlinks* que servem de material de apoio ao *card* em discussão, pode ser utilizado tanto para fins de embasamento teórico, fornecendo mais material ao entendimento do *card*, quanto a uma aplicação prática do conteúdo.

Como podemos perceber, este é um arquivo JavaScript composto puramente por texto, classes e sub-classes, todos com *strings* como valores. Na variável *motivation*, para se criar os parágrafos de discussão dos *cards* é preciso separar linha a linha, por limitação inicial da ferramenta. Para a variável *links*, temos um vetor de matriz para criar os *hyperlinks* de referência com apenas o campo da descrição.

Agora, veremos em detalhes mais aprofundados como se faz para editar e criar os *cards* no guia.

3.5 Editando os *cards*

A seguir, veremos como utilizar o guia para os tópicos abaixo:

- Como inserir uma nova classe para um novo *card*.
- Como inserir um novo *card* para classe já existente.
- Como editar um *card*.

Para o prosseguimento destas informações, é necessário que o usuário possua um nível básico de entendimento em lógica de programação e um editor de texto. Conhecimentos em HTML, CSS e JavaScript podem ajudar nesta etapa, porém para fins didáticos estaremos criando do zero dois novs *cards*, editando um deles para uma classe já existente e explicando como inserir ou retirar informações presentes neste *card* (brevemente descrito na seção 3.4.5).

3.5.1 Como editar uma nova classe para uma novo *card*

Para a criação de uma nova classe para a inserção de uma novo *card*, é necessário seguir os passos abaixo:

Inserir nome da classe no arquivo *game.html*

Para inserir o nome da classe no arquivo *game.html*, é necessário abrir o arquivo com um editor de texto. Após abrir o arquivo, procurar pela variável "div" *container filters* (linha 29 do código fonte original), copiar uma das linhas dentro da classe *"select name="filterCards* e inserir com o *value* e nome para exibição. Para este exemplo, estaremos inserindo a classe *Trustworthiness*.

Listing 3.24: Classe *Trustworthiness* inserida na linha 14

```
1  <section class="container__options container">
2          <div class="container__filters">
3            <label for="filterCards" class="filter__paragraph">
                Filters:</label>
4            <select name="filterCards" id="filterCards">
5              <option value="all">All</option>
6              <option value="Analyze">Analyze</option>
7              <option value="Transparency">Transparency</option>
8              <option value="Safety & Security">Safety &
                  Security</option>
9              <option value="Fairness">Fairness</option>
10             <option value="Data">Data</option>
11             <option value="Agency & Oversight">Agency &
                  Oversight</option>
```

```
12              <option value="Wellbeing">Wellbeing</option>
13              <option value="Accountability">Accountability</
                    option>
14              <option value="Trustworthiness">Trustworthiness</
                    option>
15            </select>
16          </div>
```

Inserir nome da classe no arquivo *game.css*

Após a inserção, temos a classe nova aparecendo no menu dropdown de filtragem, porém o *card* ainda não foi criado. Para a criação do *card*, agora é necessário prosseguir para o arquivo *game.css* e inserir o CSS da nova classe, que é o que dará a cor ao tipo da classe no guia através do valor *background-color*. Para a seleção, basta inserir o valor hexadecimal da cor, conforme no snippet abaixo. Lembrando que, por limitação do CSS, para este campo é preciso que o valor esteja todo em minúsculo, e caso o nome seja composto deve estar separado com o uso do carácter "-" (por exemplo, se o nome do valor é "*Trust & Security*", no arquivo CSS o valor deve estar escrito como "*.trust-security*").

Listing 3.25: Inserindo o CSS para a nova classe.

```
1 .trustworthiness{
2   background-color: #31e221;
3 }
```

Inserir dados no arquivo *infoCards.js*

Após realizados estes dois passos anteriores, ao inserir as informações do *card* no arquivo *infoCards.js*, o mesmo já estará aparecendo sem maiores problemas no guia, conforme imagem abaixo.

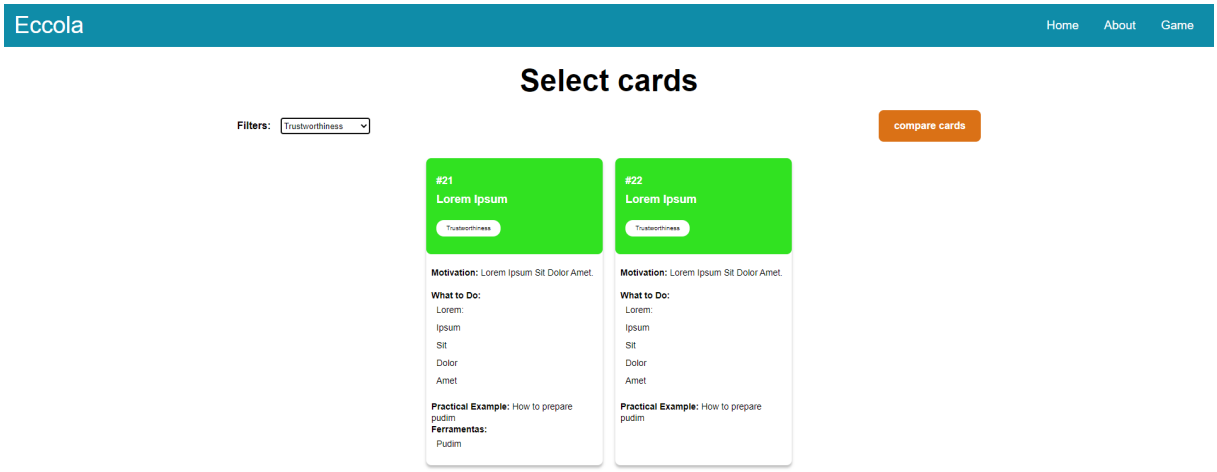

Figura 3.8: Imagem do sistema com os *cards Trustworthiness* criados.

Para realizar a inserção destas informações dos *cards*, basta colocar as informações conforme o modelo no snippet a seguir:

Listing 3.26: Informações novas inseridas nos *cards*

```
const infosCards = [
{
    number: "21",
    type: "Trustworthiness",
    title: "Lorem Ipsum",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
```

```
22    },
23
24    {
25      number: "22",
26      type: "Trustworthiness",
27      title: "Lorem Ipsum",
28      motivation:
29        "Lorem Ipsum Sit Dolor Amet.",
30      whatToDo: [
31        "Lorem:",
32        "Ipsum",
33        "Sit",
34        "Dolor",
35        "Amet",
36      ],
37      praticalExample: "How to prepare pudim",
38      links: ""
39    },
40  ]
```

3.5.2 Como editar um *card* para classe já existente

Com as informações disponibilizadas acima, caso haja algum *card* que não foi descrito conforme o requerido pelo usuário, só é necessário editar o arquivo *infosCards.js*, conforme detalhado na seção anterior. O único requisito necessário é manter a ordem de acordo com o tipo do *card* que está na variável *type*. Caso seja feita a inserção em um *card* entre duas classes já pré-existentes, será necessário tomar cuidado com a numeração, uma vez que o sistema no estado atual não se encontra com auto-ordenação implantada. Logo, ao adicionar um *card* em classe já existente, não deixar de alterar a numeração no campo *number* nos demais *cards* consequentes.

3.5.3 Como editar um *card*

Para realizar a edição de um *card* já existente, somente é necessário seguir as instruções no *snippet* 3.26, que mostra como adicionar as informações no arquivo *infoCards.js*. Para tanto, somente é necessário um editor de texto simples (por exemplo, o Bloco de Notas, presente no *Microsoft Windows* ou o VIM, presente em qualquer sistema operacional *Linux*) e salvar, sem deixar de colocar as strings entre aspas e, ao terminar de escrever o

que deseja, inserir um vírgula ao final. A inserção desta vírgula irá auxiliar inclusive se for preciso inserir novas informações no futuro sem quebrar o *layout* do *card*.

Para fins de comparaçao entre o guia ECCOLA original e o ECCOLA modificado como neste exemplo, para ver as alterações realizadas basta acessar o link `https://oggvaldo.github.com/aiethicsguide`.

3.6 Como obter o código-fonte do guia

Para a obtenção do código-fonte do guia, este se encontra localizado no GitHub com o link `https://github.com/oggvaldo/aiethicsguide`. Ao acessar, o usuário tem a opção de realizar um *fork* do projeto, puxando e mantendo os arquivos no GitHub de uso do usuário, ou baixando os arquivos selecionando o botão *Code* presente na página. Ao selecionar no botão o próprio GitHub dá a opção de baixar os arquivos como um arquivo .zip ou realizar o clone via linha de comando usando o programa Git ou os programas GitHub Desktop ou GitHub CLI (que executa a mesma função que o Git, porém com códigos mais específicos para esta plataforma).

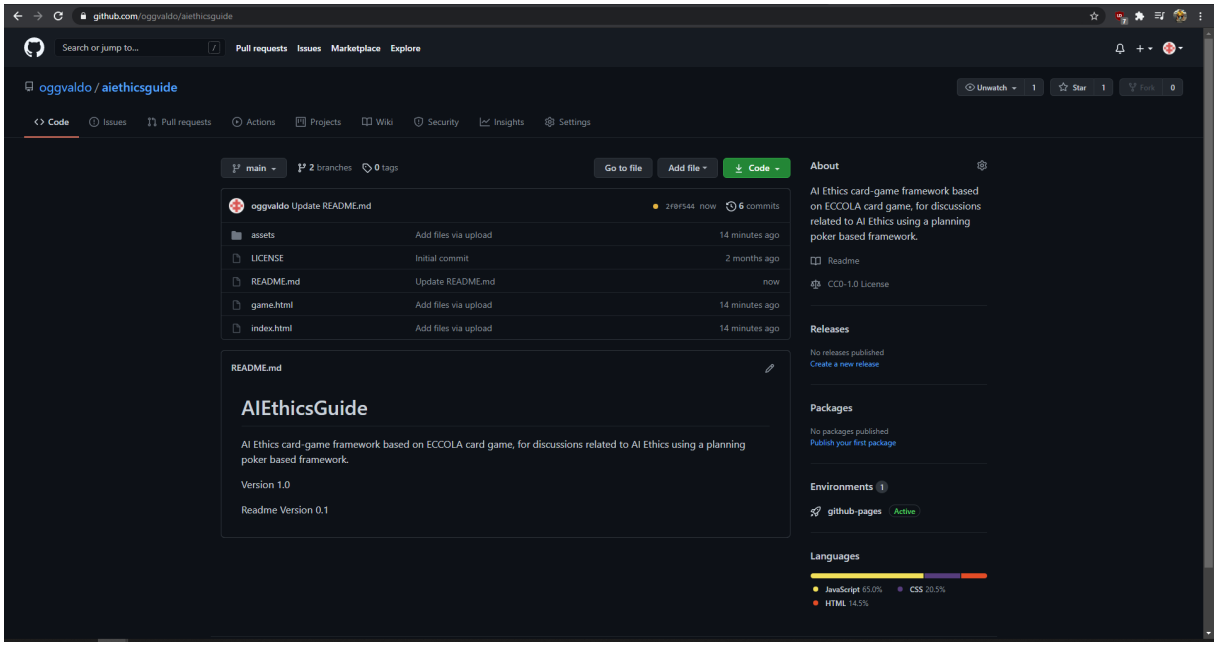

Figura 3.9: Imagem do github contendo os códigos-fontes.

3.7 Síntese do Capítulo

Foi apresentado neste Capítulo como se deu o desenvolvimento do sistema para o guia. Com a possibilidade de modularização, edição e simplicidade de uso, tanto para o usuário final – que utiliza os *cards* –, quanto ao usuário que deseja inserir seus próprios princípios éticos de IA, este guia oferece aos usuários um meio favorável para a implementação de ética em IA em projetos que envolvem o desenvolvimento de sistemas baseados em IA. Vimos a necessidade do uso de JavaScript e CSS para garantir o aspecto gráfico a este sistema baseado em HTML, o que assim permite que ele seja ao mesmo tempo um WebApp simples e prático. Também foi apresentada a construção do sistema, com uma breve documentação do código-fonte, destacando os trechos mais importantes e comentando como é realizada a edição de trechos do código para a inserção e edição de *cards*. Verificamos também algumas limitações do sistema, e como mitigá-los a fim de reduzir impactos gerados durante a etapa de inserção de novos princípios éticos, a serem empregados em versões futuras do guia.

Capítulo 4

Considerações Finais

Este trabalho teve como objetivo a criação de um protótipo no estilo de um *Planning Poker*, para permitir a discussão de princípios éticos de IA, resultando na criação deste WebApp que pode ser acessado por qualquer pessoa. Para uma prova de conceito, este sistema foi construído utilizando o guia ECCOLA, desenvolvido por Vakkuri et al. [1], uma vez que possui princípios bem descritos e foi desenvolvido desde o seu início para ser utilizado como um *Planning Poker*.

Esperamos que com esta ferramenta os desenvolvedores, em especial aqueles voltados para a área de IA e *Machine Learning* (uma vez que esta área também se baseia em princípios da IA para o desenvolvimento de suas ferramentas), possam aproveitar deste guia para não apenas criar em cima de princípios éticos já sugeridos, como os citados por Ryan e Stahl [7] ou definidos e estabelecidos como os usados pela IEEE [50], mas também para que possam criar e definir o desenvolvimento em IA com uma importante discussão em cima dos princípios éticos, fundamentais para o bom estabelecimento do código e suas diretrizes.

Mesmo em sua versão inicial, o guia já apresenta robustez no uso para a discussão de tais princípios com um *Scrum Master* ou *Product Owner* apresentando os *cards* por meio digital (uma vez que a pandemia de Covid-19 popularizou este modelo de reunião) e seus demais desenvolvedores discutindo com auxílio dos *cards* em uma videochamada. Assim, com este modelo inicial, temos também em mente possíveis melhoras para as próximas iterações que podem vir a ser implantadas em trabalhos futuros.

4.1 Trabalhos futuros

Para trabalhos futuros nós apontamos a possibilidade de melhorias na interface gráfica do sistema, deixando-o ainda mais limpo e editável através da própria página inicial, sendo necessários apenas alguns cliques e paciência por parte do *Product Owner* inserir

os dados necessários para a criação de novos *cards* sem a necessidade de alterações diretas no código-fonte da aplicação.

Outra possível melhora que traria impacto imediato seria a implantação de um *back-end* robusto, com conexão persistente, a fim de viabilizar a implementação do modo cliente-servidor, dessa forma, não haveria a necessidade de que alguma pessoa no cargo de *Product Owner* ou *Scrum Master* esteja mostrando a tela enquanto os cards são escolhidos. Com a implantação deste *back-end* poderíamos ter as cartas sendo selecionadas e mostradas através de uma videochamada, mas também permitindo a interatividade no próprio navegador com os participantes da reunião de planejamento da *sprint*.

Devido o tempo para a implementação do guia, verificamos também que o menu de filtragem dos *cards* não se encontra plemanemte funcional, permitindo apenas mostrar menos *cards* na tela, e não selecioná-los em várias filtragens consecutivas. Isto poderá ser resolvido implementando uma função para se armazenar em *buffer* de memória os cards já selecionados, mesmo com o filtro ativo em outras categorias.

Por fim, apontamos a implantação de um banco de dados para armazenar *templates* de requisitos éticos já existentes e armazenamento dos requisitos éticos elaborados pelos desenvolvedores que melhor se encaixam em seu projeto, permitindo assim uma ferramenta que pode ser usada *out-of-the-box* sendo necessário somente executar e já iniciar o uso sem grandes edições como nesta demonstração.

Isto permite o acesso e aproveitamento mesmo por usuários com pouco ou nenhum conhecimento em lógica de programação e das linguagens utilizadas a momento neste guia (JavaScript, HTML e CSS) com poucos cliques.

Referências

[1] Vakkuri, Ville, Kai-Kristian Kemell e Pekka Abrahamsson: *ECCOLA - a method for implementing ethically aligned AI systems*. Em *46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020, Portoroz, Slovenia, August 26-28, 2020*, páginas 195–204. IEEE, 2020. https://doi.org/10.1109/SEAA51224.2020.00043. v, vi, x, 1, 2, 3, 4, 19, 20, 25, 31, 32, 53

[2] Sommerville, Ian: *Software engineering, 8th Edition*. International computer science series. Addison-Wesley, 2007, ISBN 9780321313799. https://www.worldcat.org/oclc/65978675. ix, 1, 2, 7, 8, 9, 10, 11

[3] Deemer, Pete, Gabrielle Benefield, Craig Larman e Bas Vodde: *The scrum primer*, 2012. https://scrumprimer.org/scrumprimer20_small.pdf, acesso em 2021-28-03. ix, 13

[4] Paradigm, Visual: *What is planning poker in agile?*, 2020. https://www.visual-paradigm.com/scrum/what-is-agile-planning-poker/. ix, 14

[5] Jeff Sutherland, Ken Schwaber: *Scrum development process*, 1997. http://www.jeffsutherland.org/oopsla/schwapub.pdf, acesso em 2021-11-03. x, 11, 12, 13

[6] Russell, Stuart J. e Peter Norvig: *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education, 2010. x, 15, 16

[7] Ryan, Mark e Bernd Carsten Stahl: *Artificial intelligence ethics guidelines for developers and users: clarifying their content and normative implications*. Journal of Information, Communication and Ethics in Society, 2020. x, 18, 24, 31, 53

[8] Vogelsang, Andreas e Markus Borg: *Requirements engineering for machine learning: Perspectives from data scientists*. Em *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, páginas 245–251. IEEE, 2019. 1

[9] Morley, Jessica, Luciano Floridi, Libby Kinsey e Anat Elhalal: *From what to how: An initial review of publicly available ai ethics tools, methods and research to translate principles into practices*. Science and Engineering Ethics, páginas 1–28, 2019. 1, 2, 3, 19

[10] Vakkuri, Ville, Kai Kristian Kemell, Joni Kultanen e Pekka Abrahamsson: *The current state of industrial practice in artificial intelligence ethics*. IEEE Software, 2020. 1, 2, 3

[11] The Guardian: *A beauty contest was judged by AI and the robots didn't like dark skin*, setembro 2016. https://www.theguardian.com/technology/2016/sep/08/artificial-intelligence-beauty-contest-doesnt-like-black-people, acesso em 2021-04-26. 1

[12] The Verge: *Twitter taught Microsoft's AI chatbot to be a racist asshole in less than a day*, março 2016. https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist, acesso em 2021-04-26. 1

[13] Mittelstadt, Brent: *Principles alone cannot guarantee ethical AI.* Nature Machine Intelligence, 1(11):501–507, novembro 2019. https://doi.org/10.1038/s42256-019-0114-4. 1

[14] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland e Dave Thomas: *Manifesto for agile software development*, 2001. http://agilemanifesto.org/, acesso em 2021-15-03. 1, 11

[15] Morley, Jessica, Anat Elhalal, Francesca Garcia, Libby Kinsey, Jakob Mokander e Luciano Floridi: *Ethics as a service: a pragmatic operationalisation of AI ethics.* CoRR, abs/2102.09364, 2021. https://arxiv.org/abs/2102.09364. 2, 3

[16] Kostova, Blagovesta, Seda Gurses e Alain Wegmann: *On the interplay between requirements, engineering, and artificial intelligence.* Em *Proceedings of REFSQ-2020 Workshops, Pisa, Italy, 2020*, páginas 1–5, 2020. http://ceur-ws.org. 2

[17] Siqueira De Cerqueira, José Antonio, Lucas Dos Santos Althoff, Paulo Santos De Almeida e Edna Dias Canedo: *Ethical perspectives in ai: A two-folded exploratory study from literature and active development projects.* Em *Proceedings of the 54th Hawaii International Conference on System Sciences*, página 5240, 2021. 2, 3, 19

[18] Leslie, David: *Understanding artificial intelligence ethics and safety.* arXiv preprint arXiv:1906.05684, 2019. 3

[19] Jobin, Anna, Marcello Ienca e Effy Vayena: *The global landscape of ai ethics guidelines.* Nature Machine Intelligence, 1(9):389–399, 2019. 3, 18

[20] Vaishnavi, Vijay K. e William Kuechler: *Design Science Research Methods and Patterns: innovating information and communication technology.* CRC Press, 2015. 4

[21] Copeland, B. Jack: *Colossus: Its origins and originators.* IEEE Ann. Hist. Comput., 26(4):38–45, 2004. https://doi.org/10.1109/MAHC.2004.26. 6

[22] Dijkstra, Edsger W.: *The humble programmer.* Commun. ACM, 15(10):859–866, 1972. https://doi.org/10.1145/355604.361591. 6

[23] Galler, Bernard A.: *ACM president's letter: NATO and software engineering?* Commun. ACM, 12(6):301, 1969. https://doi.org/10.1145/363011.363013. 7

[24] Pressman, Roger: *Software Engineering: A Practitioner's Approach, 7th edition*. McGraw Hill, 2010, ISBN 9780073375977. http://sendy.itmaranatha.org/PSI/RPL-7th_ed_software_engineering_a_practitioners_approach_by_roger_s._pressman_.pdf. 7, 8, 10, 12

[25] Wazlawick, Raul Sidnei: *Engenharia de Software: Conceitos e Práticas, 2ª edição*. GEN LTC, 2019, ISBN 9788535292725. 7

[26] Fleming, Ian: *Cmmi - requirements development (rd) process area*, 2004. https://www.software-quality-assurance.org/cmmi-requirements-development.html, acesso em 2021-05-03. 9

[27] Hadar, Irit, Pnina Soffer e Keren Kenzi: *The role of domain knowledge in requirements elicitation via interviews: an exploratory study*. Requir. Eng., 19(2):143–159, 2014. https://doi.org/10.1007/s00766-012-0163-2. 10

[28] Bourque, Pierre e Richard E. Fairley (editores): *SWEBOK: Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, Los Alamitos, CA, version 3.0 edição, 2014, ISBN 978-0-7695-5166-1. http://www.swebok.org/. 10

[29] HE:Labs: *Desenvolvimento Ágil*, 2014. http://www.desenvolvimentoagil.com.br/scrum/, acesso em 2021-26-03. 11

[30] Grenning, James: *Planning poker or how to avoid analysis paralysis while release planning*, 2002. https://wingman-sw.com/papers/PlanningPoker-v1.1.pdf, acesso em 2021-18-03. 13

[31] Wanderley, Eduardo Garcia, Alexandre Vasconcelos e Bruno Tenório Ávila: *Using function points in agile projects: A comparative analysis between existing approaches*. Em *WBMA*, volume 802 de *Communications in Computer and Information Science*, páginas 47–59. Springer, 2017. 13

[32] Haugen, Nils Christian: *An empirical study of using planning poker for user story estimation*. Em *AGILE*, páginas 23–34. IEEE Computer Society, 2006. 13

[33] Binhamid, Abdulelah, Taghi Javdani Gandomani e Tieng Wei Koh: *"a case study research on software cost estimation using experts' estimates, wideband delphi, and planning poker technique*. International Journal of Software Engineering and Its Applications, 8(11):173–182, 2004. 14

[34] Mahnic, Viljan e Tomaz Hovelja: *On using planning poker for estimating user stories*. J. Syst. Softw., 85(9):2086–2095, 2012. 14

[35] Moløkken-Østvold, Kjetil, Nils Christian Haugen e Hans Christian Benestad: *Using planning poker for combining expert estimates in software projects*. J. Syst. Softw., 81(12):2106–2117, 2008. 14

[36] Abrahamsson, Pekka, Ilenia Fronza, Raimund Moser, Jelena Vlasenko e Witold Pedrycz: *Predicting development effort from user stories*. Em *ESEM*, páginas 400–403. IEEE Computer Society, 2011. 15

[37] Gandomani, Taghi Javdani, Hamidreza Faraji e Mahsa Radnejad: *Planning poker in cost estimation in agile methods: Averaging vs. consensus*. IEEE 5th International Conference on Knowledge-Based Engineering and Innovation (KBEI) (Iran University of Science and Technology), páginas 66–71, 2019. 15

[38] Haugeland, John: *Artificial intelligence - the very idea*. MIT Press, 1989, ISBN 978-0-262-58095-3. 15

[39] Stalder, Felix: *The age of spiritual machines: When computers exceed human intelligence*. SIGCAS Comput. Soc., 31(1):23–25, 2001. https://doi.org/10.1145/572277.572284. 15

[40] Charniak, Eugene e Robert P. Goldman: *A bayesian model of plan recognition*. Artif. Intell., 64(1):53–79, 1993. https://doi.org/10.1016/0004-3702(93)90060-O. 15

[41] Poole, David, Alan K. Mackworth e Randy Goebel: *Computational intelligence - a logical approach*. Oxford University Press, 1998, ISBN 978-0-19-510270-3. 15

[42] Stair, Ralph M. e George W. Reynolds: *Principles of Information Systems, A Managerial Approach, 9th edition*. Cenage Learning, 2010. 16

[43] Kaufman, Dora: *Inteligencia artificial: Questoes eticas a serem enfrentadas*. Anais Eletrônicos, 2016. http://abciber.org.br/anaiseletronicos/wp-content/uploads/2016/trabalhos/inteligencia_artificial_questoes_eticas_a_serem_enfrentadas_dora_kaufman.pdf. 16, 17

[44] SIMONITE, TOM: *Inteligência artificial arruinou o xadrez; agora, ela está tornando o jogo belo novamente*. Wired Festival Brasil, 2020. https://wiredfestival.globo.com/Ideias/noticia/2020/10/inteligencia-artificial-arruinou-o-xadrez-agora-ela-esta-tornando-o-jogo-belo-nov
html, acesso em 2021-03-31. 17

[45] Santino, Renato: *Campeão de go se aposenta após ser derrotado por inteligência artificial do google*. Olhar Digital, 2019. https://olhardigital.com.br/2019/11/27/seguranca/campeao-de-go-se-aposenta-apos-ser-derrotado-por-inteligencia-artificial-do-googl
acesso em 2021-03-31. 17

[46] Fisher, Josie: *Social responsibility and ethics: Clarifying the concepts*. Journal of Business Ethics, 52(4):391–400, 2004. http://www.jstor.org/stable/25123269. 17

[47] Beauchamp, Tom L., Denis G. Arnold e Norman E. Bowie: *Ethical Theory and Business*. Pearson, 2012. 17

[48] Guizzardi, Renata S. S., Glenda Carla Moura Amaral, Giancarlo Guizzardi e John Mylopoulos: *Ethical requirements for AI systems*. Em *Canadian Conference on AI*, volume 12109 de *Lecture Notes in Computer Science*, páginas 251–256, https://doi.org/10.1007/978-3-030-47358-7_24, 2020. Springer. 18

[49] European Commission: *Ethics Guidelines for Trustworthy AI High-Level Expert Group on artificial intelligence*, abril 2019. https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai, acesso em 2020-10-08. 18, 19, 20

[50] IEEE: *Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems (EAD FirstEdition)*, 2019. https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/other/ead1e.pdf, acesso em 2020-10-08. 18, 19, 20, 53

[51] Hagendorff, Thilo: *The ethics of AI ethics: An evaluation of guidelines*. Minds Mach., 30:99–120, 2020. 18

[52] Fjeld, Jessica, Nele Achten, Hannah Hilligoss, Adam Nagy e Madhulika Srikumar: *Principled artificial intelligence: Mapping consensus in ethical and rights-based approaches to principles for AI*. SSRN Electronic Journal, 1(1):1–39, 2020. https://doi.org/10.2139/ssrn.3518482. 18

[53] Zeng, Y., Enmeng Lu e Cunqing Huangfu: *Linking artificial intelligence principles*. ArXiv, abs/1812.04814, 2019. 18

[54] Smit, Koen, Martijn Zoet e John van Meerten: *A review of AI principles in practice*. Em Vogel, Doug, Kathy Ning Shen, Pan Shan Ling, Carol Hsu, James Y. L. Thong, Marco De Marco, Moez Limayem e Sean Xin Xu (editores): *24th Pacific Asia Conference on Information Systems, PACIS 2020, Dubai, UAE, June 22-24, 2020*, página 198, 2020. https://aisel.aisnet.org/pacis2020/198. 18

[55] Consortium, World Wide Web: *HTML - Living Standard*, abril 2021. https://html.spec.whatwg.org/multipage/. 22

[56] Bert Bos, World Wide Web Consortium: *CSS*, abril 2021. https://www.w3.org/Style/CSS/. 22

[57] Docs, Mozilla MDN Web: *JavaScript*, janeiro 2021. https://developer.mozilla.org/en-US/docs/Web/JavaScript. 22

[58] Nations, Daniel: *What is a web application?* Lifewire, 2020. https://www.lifewire.com/what-is-a-web-application-3486637, acesso em 2021-04-09. 23

[59] Google: *Material Design*, janeiro 2021. https://material.io/design. 25

Apêndice A

Imagens da interface do programa

Figura A.1: Imagem do sistema adaptado para telas de computadores de mesa e *notebooks*.

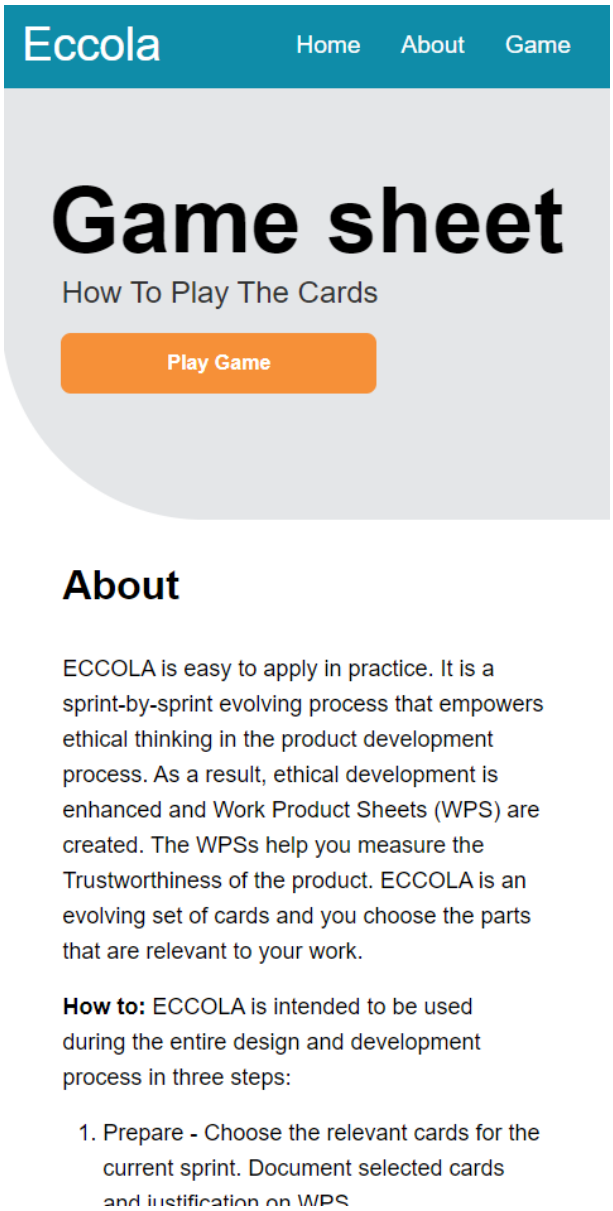

Figura A.2: Imagem do sistema adaptado para telas de *smartphones* e *tablets*

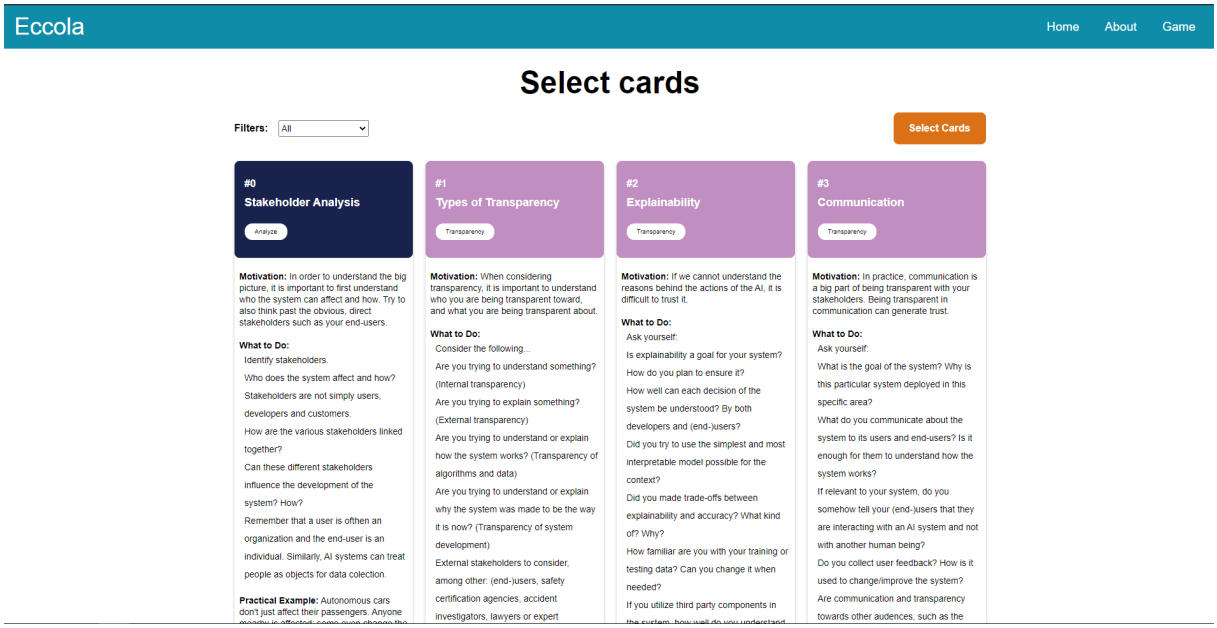

Figura A.3: Imagem do protótipo do sistema na tela inicial criado na ferramenta Figma.

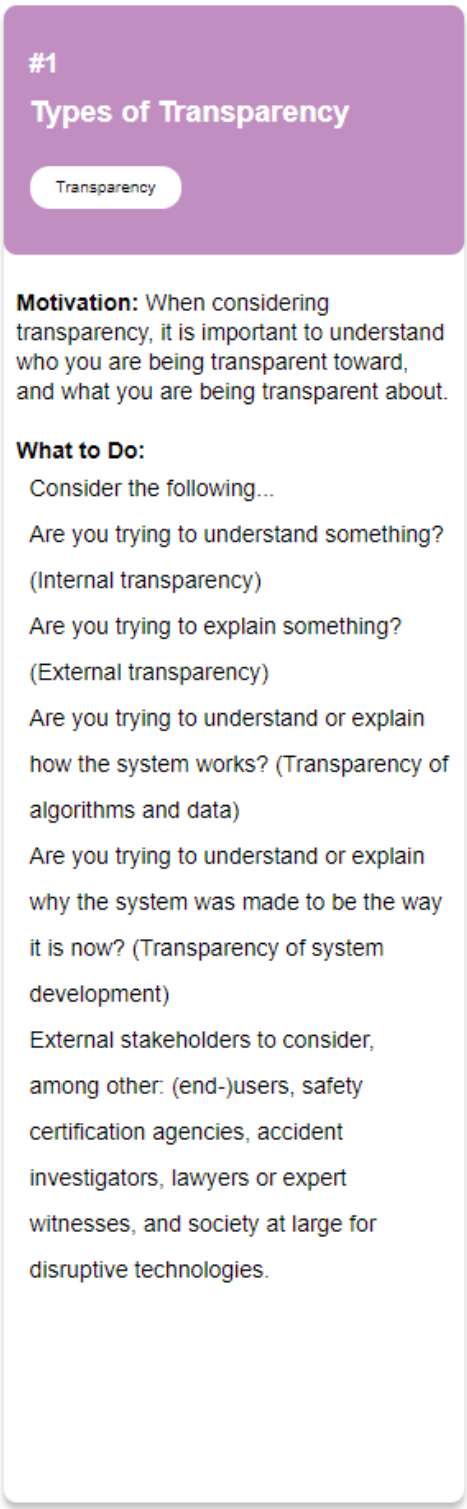

Figura A.4: Detalhe de um card do Eccola na versão final para computadores de mesa.

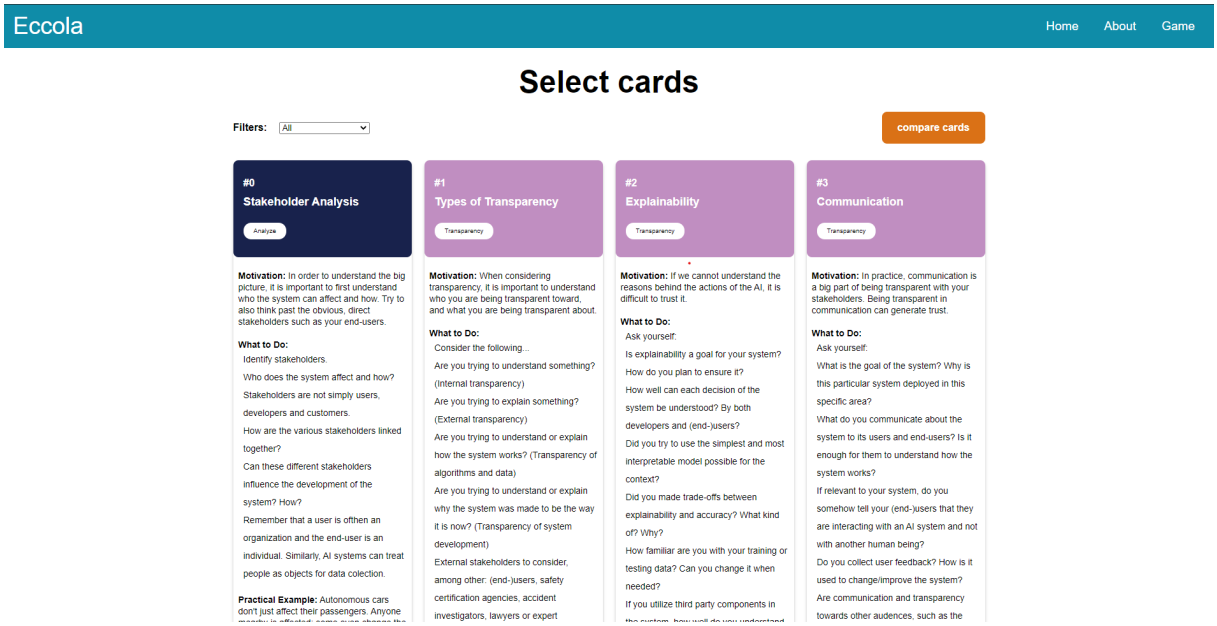

Figura A.5: Imagem do sistema adaptado para telas de computadores de mesa e *notebooks* na tela de seleção de cards.

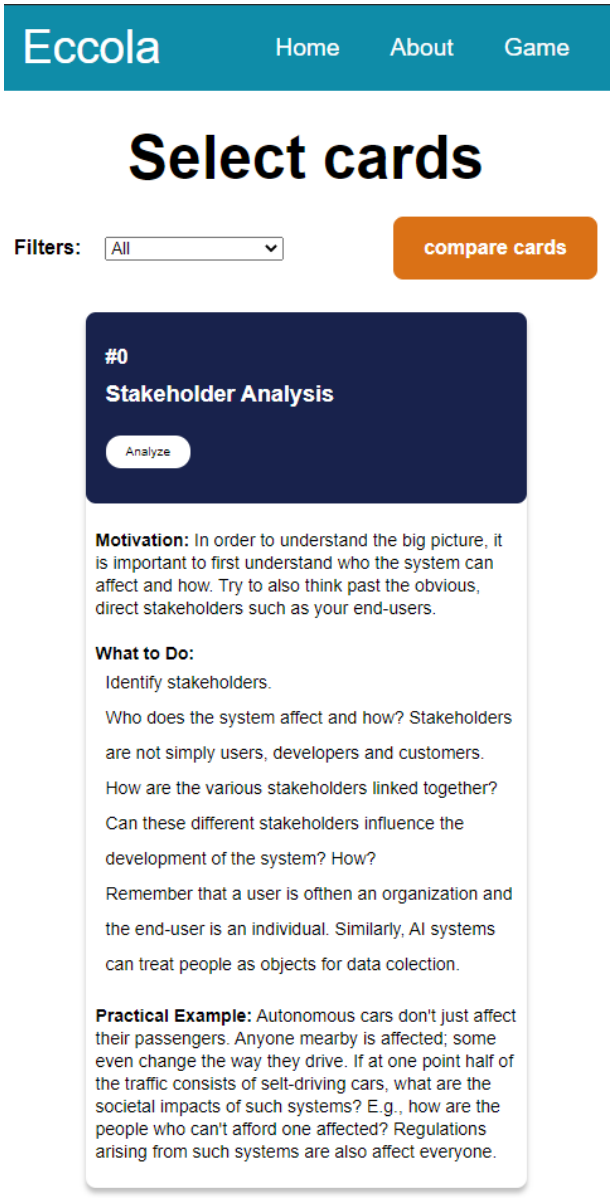

Figura A.6: Imagem do sistema adaptado para telas de celulares e *tablets* e *notebooks* na tela de seleção de cards.

Apêndice B

Código-fonte do guia

B.1 *index.html*

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial
       -scale=1.0" />
    <title>AI Ethics Guide</title>
    <link rel="stylesheet" href="./assets/css/reset.css" />
    <link rel="stylesheet" href="./assets/css/style.css" />
  </head>
  <body>
    <header class="header">
      <!-- Nav -->
      <nav class="nav">
        <a  href="index.html" class="nav__logo">AI Ethics
           Guide</a>
        <ul class="menu">
          <li class="menu__item"><a href="index.html">Home</a>
             </li>
          <li class="menu__item"><a href="#Whatisit">What is
             it</a></li>
          <li class="menu__item"><a href="game.html">Game</a><
             /li>
```

```
20          <li class="menu__item"><a href="about.html">About</a
                ></li>
21        </ul>
22      </nav>
23      <!-- Secao principal -->
24      <section class="principal">
25        <div class="content__title">
26          <h1 class="title">Game sheet</h1>
27          <h3 class="subtitle">How To Play The Cards</h3>
28          <a href="game.html" class="button__game">Play Game</
                a>
29        </div>
30        <div>
31          <picture>
32            <img
33              class="illustration"
34              src="./assets/img/card-game.svg"
35              alt="Imagem de uma mao segurando tres cartas,
                    uma na cor azul escuro, laranja e azul claro"
36            />
37          </picture>
38        </div>
39      </section>
40    </header>
41    <!-- Secao About -->
42    <main class="container main">
43      <h2 class="about__title" id="Whatisit">What is it</h2>
44      <p class="about__paragraph">
45        Lorem ipsum dolor sit amet, consectetur adipiscing
             elit. Integer gravida tortor id blandit aliquam.
             Quisque dapibus semper tristique. Aliquam sem est,
             mollis at euismod ut, aliquam a libero. Morbi
             porttitor vel sapien in ultrices. Nam tempus
             consequat vestibulum. Nullam ac nulla suscipit,
             semper felis sit amet, vulputate dui. Aliquam et
             sapien nulla. Nunc lacinia purus ut ex pretium
             varius. Pellentesque ornare convallis dolor id
             malesuada.
46      </p>
47      <p class="about__paragraph">
```

```
48          Morbi nulla magna, ultricies vel facilisis et, congue
               ut dolor. Duis eu tempor neque. Nam lorem nisi,
               ornare in nulla et, accumsan dapibus turpis. Cras
               sed magna ac neque fermentum tincidunt. Mauris
               eleifend tempor porta. Vivamus vestibulum rutrum
               semper. Sed ullamcorper metus blandit ante rutrum,
               eget blandit lorem faucibus. Nulla dapibus
               porttitor suscipit. Pellentesque ac tempor nisi, a
               fermentum risus. Fusce convallis lorem id ipsum
               lobortis elementum. Sed varius magna at mi luctus,
               fringilla venenatis lectus volutpat.
49        </p>
50        <p class="about__paragraph">
51          Aliquam at semper sapien, nec malesuada dui. Ut ante
               sem, ultrices vitae nibh vitae, tempus pulvinar est
               . Donec malesuada, turpis sit amet faucibus
               venenatis, nulla neque consequat lacus, id
               condimentum ligula diam fringilla urna. Ut eu
               fermentum justo. Sed non sapien auctor, euismod
               lorem in, cursus tellus. Sed eget molestie massa,
               id hendrerit ex. Ut laoreet sem eget massa bibendum
                tincidunt. Donec vestibulum dolor condimentum elit
                posuere vehicula. Morbi et velit nulla. Quisque
               convallis lectus vel turpis tristique molestie.
               Aenean a porttitor sapien. Mauris id venenatis
               turpis, vehicula faucibus neque. Aliquam suscipit
               malesuada bibendum. Sed a nibh at ex interdum
               laoreet ac sit amet elit. Integer dapibus eu diam
               sit amet mollis.
52        </p>
53        <p class="about__paragraph">
54          Sed a semper odio. Sed non nulla magna. Aliquam
               iaculis tincidunt nulla. Phasellus dui neque,
               tincidunt eu luctus eu, facilisis a enim.
               Suspendisse ac molestie erat. Maecenas pretium in
               nunc vitae semper. Curabitur sed eros ac arcu
               volutpat consequat vitae eget enim. Orci varius
               natoque penatibus et magnis dis parturient montes,
               nascetur ridiculus mus. Nunc id viverra sapien, nec
                mattis est. Duis semper pellentesque metus, sed
```

```
          hendrerit est commodo id. Donec iaculis, libero id
          scelerisque consequat, justo nisl vestibulum ex, id
           auctor orci ligula vel metus. Integer fermentum
          ipsum porta dui feugiat, in hendrerit dui faucibus.
           Nulla nisi justo, egestas ac ante ut, viverra
          vestibulum sem. Praesent tempor pretium velit non
          convallis. Nulla vestibulum arcu vitae fermentum
          tincidunt.
55      </p>
56      <p class="about__paragraph">
57        Curabitur lacinia mattis auctor. Donec nibh eros,
             pellentesque a nunc eget, consectetur finibus
             ligula. In a elementum leo, non suscipit purus.
             Suspendisse massa nisi, ultrices eu lorem a,
             iaculis feugiat massa. Quisque eget tortor et dolor
              ornare dictum non a magna. Curabitur convallis
             velit vitae tellus sodales, sit amet suscipit
             mauris tincidunt. In hac habitasse platea dictumst.
              Proin congue in massa eu dignissim.
58      </p>
59      <div class="explication_container">
60        <div class="explication">
61          <p class="about__paragraph">
62            <span class="strong">How to:</span> To play this
                 game is simple:
63          </p>
64          <ol start="1" class="about__list">
65            <li>
66              Prepare: Choose the relevant cards for the
                   current sprint.
67            </li>
68            <li>
69              Review: Keep the selected cards on hand during
                   single tasks.
70            </li>
71            <li>
72              Evaluate: Review to ensure that all planned
                   actions are taken.
73            </li>
74          </ol>
```

```
75          </div>
76          <div class="explication_img">
77            <picture>
78              <a href='https://br.freepik.com/vetores/negocio'>
79                <img src="./assets/img/19963.svg" alt="Negocio
                     vetor criado por katemangostar - br.freepik.
                     com/vetores/negocio" srcset="" />
80                <p><b href='https://br.freepik.com/vetores/
                     negocio'>Negocio vetor criado por
                     katemangostar - br.freepik.com</b></p>
81              </a>
82            </picture>
83          </div>
84          <p class="about__paragraph">
85            Lorem ipsum dolor sit amet, consectetur adipiscing
                 elit, sed do eiusmod tempor incididunt ut labore
                 et dolore magna aliqua. Ut enim ad minim veniam,
                 quis nostrud exercitation ullamco laboris nisi ut
                  aliquip ex ea commodo consequat. Duis aute irure
                  dolor in reprehenderit in voluptate velit esse
                 cillum dolore eu fugiat nulla pariatur. Excepteur
                  sint occaecat cupidatat non proident, sunt in
                 culpa qui officia deserunt mollit anim id est
                 laborum.
86          </p>
87        </div>
88      </main>
89      <!-- Secao Footer -->
90      <footer class="footer">
91        <a href="game.html" class="button__game">Play Game</a>
92      </footer>
93    </body>
94  </html>
```

B.2 *game.html*

```
1     <!DOCTYPE html>
2   <html lang="pt-BR">
3     <head>
```

```
4     <meta charset="UTF-8" />
5     <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6     <meta name="viewport" content="width=device-width, initial
        -scale=1.0" />
7     <title>Game Play</title>
8     <link rel="stylesheet" href="./assets/css/reset.css" />
9     <link rel="stylesheet" href="./assets/css/style.css" />
10    <link rel="stylesheet" href="./assets/css/game.css" />
11  </head>
12  <body>
13    <header class="header">
14      <!-- Nav -->
15      <header class="header">
16        <!-- Nav -->
17        <nav class="nav">
18          <a  href="index.html" class="nav__logo">AI Ethics
              Guide</a>
19          <ul class="menu">
20            <li class="menu__item"><a href="index.html">Home</
                a></li>
21            <li class="menu__item"><a href="index.html#
                Whatisit">What is it</a></li>
22            <li class="menu__item"><a href="game.html">Game</a
                ></li>
23            <li class="menu__item"><a href="about.html">About<
                /a></li>
24          </ul>
25        </nav>
26    </header>
27    <!-- Cards -->
28    <main>
29      <div class="container__game container">
30        <h1 id="game__title">Select cards</h1>
31      </div>
32      <section class="container__options container">
33        <div class="container__filters">
34          <label for="filterCards" class="filter__paragraph">
              Filters:</label>
35          <select name="filterCards" id="filterCards">
36            <option value="all">All</option>
```

```
37              <option value="Template1">Template 1</option>
38              <option value="Template2">Template 2</option>
39              <option value="Template3 & Template4">Template 3 &
                    Template 4</option>
40              <option value="Template5">Template 5</option>
41              <option value="Template6">Template 6</option>
42              <option value="Template7 & Template 8">Template 7
                   & Template 8</option>
43              <option value="Template9">Template 9</option>
44              <option value="Template10">Template 10</option>
45              <option value="Template11">Template 11</option>
46            </select>
47          </div>
48          <div>
49            <button id="result" class="btn__compare">Compare
                 Cards</button>
50            <button id="start" class="btn__start">Start Again</b
                 utton>
51          </div>
52        </section>
53        <section class="container">
54          <div class="container__alert">
55            <!-- alerta quando a pessoa nao seleciona pelo menos
                dois cards -->
56            <p class="alert">Select at least a card</p>
57          </div>
58          <!-- onde e criado os cards -->
59          <div class="cards"></div>
60        </section>
61      </main>
62      <script src="./assets/js/infosCards.js"></script>
63      <script src="./assets/js/index.js"></script>
64    </body>
65  </html>
```

B.3 *about.html*

```
1     <!DOCTYPE html>
2   <html lang="pt-BR">
```

```
3   <head>
4     <meta charset="UTF-8" />
5     <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6     <meta name="viewport" content="width=device-width, initial
        -scale=1.0" />
7     <title>AI Ethics Guide</title>
8     <link rel="stylesheet" href="./assets/css/reset.css" />
9     <link rel="stylesheet" href="./assets/css/style.css" />
10  </head>
11  <body>
12    <header class="header">
13      <!-- Nav -->
14      <nav class="nav">
15        <a  href="index.html" class="nav__logo">AI Ethics
            Guide</a>
16        <ul class="menu">
17          <li class="menu__item"><a href="index.html">Home</a>
              </li>
18          <li class="menu__item"><a href="#Whatisit">What is
              it</a></li>
19          <li class="menu__item"><a href="game.html">Game</a><
              /li>
20          <li class="menu__item"><a href="about.html">About</a
              ></li>
21        </ul>
22      </nav>
23      <!-- Secao principal -->
24      <section class="principal">
25        <div class="content__title">
26          <h1 class="title">Game sheet</h1>
27          <h3 class="subtitle">How To Play The Cards</h3>
28          <a href="game.html" class="button__game">Play Game</
              a>
29        </div>
30        <div>
31          <picture>
32            <img
33              class="illustration"
34              src="./assets/img/card-game.svg"
```

```
35              alt="Imagem de uma mao segurando tres cartas,
                   uma na cor azul escuro, laranja e azul claro"
36            />
37          </picture>
38        </div>
39      </section>
40    </header>
41    <!-- Secao About -->
42    <main class="container main">
43      <h2 class="about__title" id="About">About</h2>
44      <p class="about__paragraph">
45        Created by: Anayran Pinheiro
46      </p>
47      <p class="about__paragraph">
48        License for this tool: Creative Commons. Use it for
             free, just reference the license and the creator.
49      </p>
50        <p class="about__paragraph">
51          Lorem ipsum dolor sit amet, consectetur adipiscing
              elit, sed do eiusmod tempor incididunt ut labore
              et dolore magna aliqua. Ut enim ad minim veniam,
              quis nostrud exercitation ullamco laboris nisi ut
               aliquip ex ea commodo consequat. Duis aute irure
               dolor in reprehenderit in voluptate velit esse
              cillum dolore eu fugiat nulla pariatur. Excepteur
               sint occaecat cupidatat non proident, sunt in
              culpa qui officia deserunt mollit anim id est
              laborum.
52        </p>
53      </div>
54    </main>
55    <!-- Secao Footer -->
56    <footer class="footer">
57      <p><a href="http://creativecommons.org/licenses/by/4.0/"
           rel="license"><img style="border-width: 0;" src="
          https://i.creativecommons.org/l/by/4.0/80x15.png" alt
          ="Creative Commons License" /><br />
58      </a><br />
59          Unless otherwise stated, the contents are licensed
              under
```

```
60          <a href="http://creativecommons.org/licenses/by/4.0/"
               rel="license">Creative Commons Attribution 4.0
               &#8211; International</a>.
61          The authors are responsible for the choice and
               presentation of their texts on this site and for
               the opinions expressed. <a href="https://github.com
               /oggvaldo/aiethicalguide/">Source</a>.</p>
62      </footer>
63    </body>
64  </html>
```

B.4 *index.js*

```
1       const filter = document.querySelector("#filterCards");
2   const filterContainer = document.querySelector(".
       container__filters");
3   const buttonResult = document.querySelector("#result");
4   const titlePage = document.querySelector("#game__title");
5   const startAgain = document.querySelector("#start");
6   const container = document.querySelector('.container__options'
       );
7   const alert = document.querySelector('.alert')
8
9   // Funcao que seleciona os cards e ativa o botao de resultado
10  const selectedCards = () => {
11    const cards = document.querySelectorAll(".card");
12    cards.forEach((card) => {
13      card.addEventListener("click", function () {
14        this.classList.toggle("active");
15      });
16    });
17  };
18
19  // Funcao para mostrar as cartas selecionadas
20  const statePageResult = () => {
21    buttonResult.style.display = "none";
22    filterContainer.style.display = "none";
23
24    startAgain.style.display = "block"
```

```
25    container.style.display = "flex";
26    container.style.justifyContent="center";
27
28    titlePage.innerText = "Result";
29    alert.style.display = "none";
30  };
31
32  // Funcao para selecionar as cartas
33  const statePageSelect = () => {
34    buttonResult.style.display = "block";
35    titlePage.innerText = "Select cards";
36
37    startAgain.style.display = "none";
38
39    filterContainer.style.display = "flex";
40    container.style.display = "flex";
41    container.style.justifyContent="space-between";
42    alert.style.display = 'none'
43  };
44
45  // Funcao que faz os filtros dos cards
46  filter.addEventListener("click", (event) => {
47    if (event.target.value === "all") {
48      return createCards(infosCards);
49    }
50
51    statePageSelect();
52
53    const cards = filterCards(infosCards, event.target.value);
54    return createCards(cards);
55  });
56
57  // Funcao que filtra o array
58  const filterCards = (array, type) => {
59    return array.filter((infos) => {
60      return infos.type === type;
61    });
62  };
63
64  // Funcao que formata a classes de CSS
```

```
65  const formatClasse = (type) => {
66    if (type === "Template 3 & Template 4") {
67      return "template3-template4";
68    }
69
70    if (type === "Template 7 & Template 8") {
71      return "template7-template8";
72    }
73
74    return type.toLowerCase();
75  };
76
77  //  Template de List do Card
78  const templateList = (list) => {
79    return list.reduce((accumulator, actual) => {
80      return (accumulator += `
81        <li>${actual}</li>
82       `);
83    }, "");
84  };
85
86  const templateListLink = (list) => {
87    return list.reduce((accumulator, actual) => {
88      return (accumulator += `
89        <li>
90        <a href="${actual.link}">${actual.descricao}</a>
91        </li>
92       `);
93    }, "");
94  };
95
96
97  // Funcao que pega template criado e adiciona no DOM
98  const insertCardsIntoPage = (template) => {
99    const cards = document.querySelector(".cards");
100   cards.innerHTML = template;
101   // Adiciona funcionalidade de selecionar os cards
102   selectedCards();
103 };
104
```

```
105 // Funcao para a criacao de cada um dos cards
106 const createCards = (infos) => {
107   const template = infos.reduce((accumulator, actual) => {
108     return (accumulator += `
109     <article class="card">
110     <div class="card__header ${formatClasse(actual.type)}">
111       <p class="card__number">#${actual.number}</p>
112       <h3 class="card__title">${actual.title}</h3>
113       <span class="card__badge">${actual.type}</span>
114     </div>
115     <div class="card__body">
116       <p>
117         <span class="strong">Issue: </span>${actual.motivation
              }
118       </p>
119       <br>
120       <ul>
121         <span class="strong">How to think:</span>
122         <br>
123         ${templateList(actual.whatToDo)}
124       </ul>
125       <br>
126       <p>
127         ${
128           actual.praticalExample
129             ? `<span class="strong"> Examples:</span> ${actual
                    .praticalExample} `
130             : ""
131         }
132       </p>
133       <p>
134       ${
135         actual.links
136           ? `<p class="strong"> Tool(s):</p>
137              <ul>
138              ${templateListLink(actual.links)}
139              </ul>
140            `
141           : ""
142       }
```

```
143       </p>
144       </div>
145     </article>
146       `);
147     }, "");
148     insertCardsIntoPage(template);
149 };
150 
151 // Funcao que captura todos cards que estao ativos e
      transforma em um array.
152 buttonResult.addEventListener("click", () => {
153     const selectedCards = document.querySelectorAll(".active");
154     const cards = document.querySelector(".cards");
155 
156     // Definir o minimo de cartas a ser selecionado pelo guia
157     if (selectedCards.length >= 1) {
158       cards.innerHTML = "";
159       selectedCards.forEach((card) => {
160         card.classList.remove("active");
161         cards.appendChild(card);
162       });
163       statePageResult();
164       return
165     }
166     return alert.style.display = 'block'
167 
168 });
169 
170 startAgain.addEventListener("click", () => {
171     statePageSelect();
172     createCards(infosCards);
173 });
174 
175 const init = () => {
176     createCards(infosCards);
177 };
178 
179 init();
```

B.5 *infosCards.js*

```
    const infosCards = [
  {
    number: "1",
    type: "Template1",
    title: "Lorem Ipsum",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
  },

  {
    number: "2",
    type: "Template2",
    title: "Lero",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
```

```
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
  },

  {
    number: "3",
    type: "Template 3 & Template 4",
    title: "Lorem Ipsum",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
  },

  {
    number: "4",
    type: "Template 3 & Template 4",
    title: "Lorem Ipsum",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
```

```
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
  },

  {
    number: "5",
    type: "Template5",
    title: "Lorem Ipsum",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
  },

  {
    number: "6",
    type: "Template6",
    title: "Lorem Ipsum",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
```

```
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
  },

  {
    number: "7",
    type: "Template 7 & Template 8",
    title: "Lorem Ipsum",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
  },

  {
    number: "8",
    type: "Template 7 & Template 8",
```

```
    title: "Lorem Ipsum",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
  },

  {
    number: "9",
    type: "Template9",
    title: "Lorem Ipsum",
    motivation:
      "Lorem Ipsum Sit Dolor Amet.",
    whatToDo: [
      "Lorem:",
      "Ipsum",
      "Sit",
      "Dolor",
      "Amet",
    ],
    praticalExample: "How to prepare pudim",
    links: [
      {
        descricao: 'Pudim',
        link: 'http://www.pudim.com.br'
      },
    ]
  },
```

```
199
200   {
201     number: "10",
202     type: "Template10",
203     title: "Lorem Ipsum",
204     motivation:
205       "Lorem Ipsum Sit Dolor Amet.",
206     whatToDo: [
207       "Lorem:",
208       "Ipsum",
209       "Sit",
210       "Dolor",
211       "Amet",
212     ],
213     praticalExample: "How to prepare pudim",
214     links: [
215       {
216         descricao: 'Pudim',
217         link: 'http://www.pudim.com.br'
218       },
219     ]
220   },
221
222   {
223     number: "11",
224     type: "Template11",
225     title: "Lorem Ipsum",
226     motivation:
227       "Lorem Ipsum Sit Dolor Amet.",
228     whatToDo: [
229       "Lorem:",
230       "Ipsum",
231       "Sit",
232       "Dolor",
233       "Amet",
234     ],
235     praticalExample: "How to prepare pudim",
236     links: ""
237   },
238
```

```
239 ];
```

B.6 *style.css*

```
1     *,
2 *:before,
3 *:after {
4   margin: 0;
5   padding: 0;
6   box-sizing: border-box;
7 }
8
9 html {
10   scroll-behavior: smooth;
11 }
12 body {
13   box-sizing: border-box;
14   font-size: 10px;
15   font-size: 10px;
16   font-family: "Quicksand", sans-serif;
17 }
18
19 a {
20   text-decoration: none;
21   color: inherit;
22 }
23
24 b {
25   text-decoration: none;
26   color: inherit;
27   font-size: 8px;
28   text-align: center;
29 }
30
31 .strong {
32   font-weight: bold;
33 }
34
35 .container {
```

```
36    max-width: 1200px;
37    margin: 0 auto;
38    width: 100%;
39  }
40
41  .header {
42    width: 100%;
43  }
44
45  .nav {
46    display: flex;
47    justify-content: space-between;
48    align-items: center;
49    padding: 1rem;
50    background: #067437;
51  }
52
53  .nav__logo {
54    font-size: 2.3rem;
55    color: #fff;
56  }
57
58  .menu {
59    display: flex;
60    justify-content: space-around;
61  }
62
63  .menu__item {
64    margin: 0 1rem;
65    font-size: 1.2rem;
66    color: #fff;
67    padding-bottom: 0;
68  }
69
70  .menu__item:hover {
71    font-weight: bold;
72  }
73
74  .principal {
75    padding-top: 60px;
```

```
76    background: rgba(201, 206, 210, 0.5);
77    border-bottom-left-radius: 160px;
78    display: flex;
79    align-items: center;
80    justify-content: space-around;
81    width: 100%;
82  }
83
84  .content__title {
85    margin-bottom: 100px;
86  }
87
88  .title {
89    font-style: normal;
90    font-weight: bold;
91    font-size: 100px;
92    line-height: 125px;
93    color: #000000;
94  }
95
96  .subtitle {
97    padding-left: 10px;
98    font-style: normal;
99    font-weight: normal;
100   font-size: 32px;
101   line-height: 24px;
102   color: #333333;
103 }
104
105
106
107 .button__game {
108   margin: 20px 0 0 10px;
109   background: #f69038;
110   border-radius: 8px;
111   color: #fff;
112   display: flex;
113   justify-content: center;
114   align-items: center;
115   font-size: 1rem;
```

```
  font-weight: bold;
  padding: 1rem 3rem;
  max-width: 250px;
  outline: none;
}

.button__game:hover {
  background: #da7116;
}

.main {
  padding: 32px 0;
  width: 80%;
}

.about__title {
  font-style: normal;
  font-weight: bold;
  font-size: 32px;
  line-height: 40px;
  color: #000000;
  padding-bottom: 32px;
}

.about__list {
  padding-left: 2rem;
}

.about__paragraph, p,
li {
  font-size: 18px;
  line-height: 28px;
  padding-bottom: 16px;
}

.explication_container {
  display: flex;
  flex-wrap: wrap;
}

```

```
.explication,
.explication_img {
  flex: 1;
  padding-right: 8px;
  font-size: 8px;
  text-align: center;
}

.explication_img img {
  width: 80%;
  margin-left: 2rem;
}

.footer {
  background: #333333;
  display: flex;
  font-size: 12px;
  color: #fff;
  justify-content: center;
  align-items: center;
  padding: 24px;
}

@media screen and (max-width: 1200px) {
  .container {
    max-width: 1024px;
  }
}
@media screen and (max-width: 1024px) {
  .title {
    font-size: 72px;
    line-height: 90px;
  }
  .subtitle {
    font-size: 24px;
  }

  .illustration {
    width: 300px;
    height: 300px;
```

```
196       position: relative;
197       bottom: -2px;
198     }
199
200     .explication_container {
201       flex-direction: column;
202     }
203     .explication_img img {
204       height: 400px;
205     }
206
207     .container {
208       max-width: 960px;
209     }
210   }
211
212   @media screen and (max-width: 780px) {
213     .illustration {
214       display: none;
215     }
216
217     .principal {
218       justify-content: center;
219     }
220     .explication_img img {
221       height: 290px;
222     }
223
224     .container {
225       max-width: 680px;
226     }
227   }
228
229   @media screen and (max-width: 560px) {
230     .explication_img img {
231       width: 100%;
232       margin-left: 0rem;
233   }
234   }
235
```

```
@media screen and (max-width: 480px) {
  .illustration {
    display: none;
  }

  .container {
    max-width: 100%;
  }

  .nav {
    flex-direction: column;
    padding: 16px 0;
  }

  .menu {
    margin-top: 15px;
  }

  .principal {
    justify-content: center;
    align-items: center;
    border-bottom-left-radius: 0;
    padding-top: 40px;
  }

  .content__title {
    justify-content: center;
    align-items: center;
    display: flex;
    flex-direction: column;
    width: 100%;
    margin-bottom: 50px;
  }

  .subtitle {
    padding-left: 0;
  }

  .explication_img img {
    height: 200px;
```

```
    margin-bottom: 10px;
    width: 100%;
    margin-left: 0;
  }

  .about__list {
    padding-left: 1rem;
  }

  .button__game {
    margin: 0;
    margin-top: 30px;
    max-width: 230px;
  }

  .title {
    font-size: 60px;
    line-height: 75px;
  }
}

@media screen and (max-width: 360px) {
  .illustration {
    display: none;
  }

  .title {
    font-size: 50px;
    line-height: 71px;
  }
  .subtitle {
    font-size: 20px;
  }
  .button__game {
    margin: 0;
    margin-top: 15px;
    max-width: 230px;
  }
}

```

```
316 .bloco__codigo{
317  background: gray;
318  width: 300px;
319  height: 300px;
320 }
```

B.7 *game.css*

```
1      li {
2    padding: 0;
3    padding-left: 8px;
4    margin: 0;
5    font-size: 14px;
6 }
7
8 #game__title {
9    text-align: center;
10   font-style: normal;
11   font-weight: bold;
12   font-size: 48px;
13   line-height: 60px;
14   color: #000000;
15   margin-top: 24px;
16 }
17
18 .filter__paragraph {
19   font-style: normal;
20   font-weight: bold;
21   font-size: 1rem;
22   line-height: 26px;
23   margin-right: 16px;
24   color: #000000;
25 }
26
27 .btn__filter {
28   border: none;
29   background: #c4c4c4;
30   border-radius: 4px;
31   padding: 8px;
```

```
32    margin: 0 4px;
33    cursor: pointer;
34  }
35
36  .container__game {
37    display: flex;
38    justify-content: center;
39    align-items: center;
40  }
41
42  .container__options {
43    display: flex;
44    justify-content: space-between;
45    align-items: center;
46    margin: 16px auto;
47    flex-wrap: wrap;
48    padding: 0 10px;
49  }
50
51  .btn__compare,
52  .btn__start {
53    background: #f69038;
54    border-radius: 8px;
55    color: #fff;
56    display: flex;
57    justify-content: center;
58    align-items: center;
59    font-size: 1rem;
60    font-weight: bold;
61    padding: 1rem 1.5rem;
62    border: none;
63    outline: none;
64  }
65
66  .btn__compare,
67  .btn__start:hover{
68    background: #da7116;
69  }
70
71  .btn__compare:disabled{
```

```
  opacity: 0.5;
}

/* .btn__start {
  justify-content: center;
  display: flex;
  width: 100%;
} */

.template1 {
  background-color: #18224c;
}

.template2 {
  background-color: #c08ec1;
}

.template3-template4 {
  background-color: #92c192;
}

.template5 {
  background-color: #f3bf60;
}

.template6 {
  background-color: #6393c7;
}

.template7-template8 {
  background-color: #5aaead;
}

.template9 {
  background-color: #f08d53;
}

.template10 {
  background-color: #d76f66;
}
```

```
112
113 .template11 {
114   background-color: #31e221;
115 }
116
117 .cards__top {
118   margin-top: 32px;
119 }
120
121 .cards,
122 .cards__result {
123   display: flex;
124   flex-wrap: wrap;
125   justify-content: center;
126 }
127
128 .container__alert{
129   display: flex;
130   justify-content: center;
131 }
132
133 .alert{
134   display: none;
135   font-size: 1rem;
136   color: red;
137 }
138
139 .card {
140   width: 280px;
141   border-radius: 8px;
142   box-shadow: 0px 4px 4px rgba(0, 0, 0, 0.25);
143   margin: 10px;
144 }
145
146 .card:hover {
147   animation: bounce 0.5s linear;
148 }
149
150 .card__header {
151   border-radius: 8px;
```

```
  padding: 16px;
}

.card__title {
  font-size: 18px;
  color: white;
  font-weight: bold;
  margin: 16px 0;
}

.card__badge {
  background-color: white;
  border-radius: 12px;
  padding: 8px 16px;
  display: inline-flex;
}

.card__body {
  border-top: none;
  padding-top: 20px;
  padding: 20px 8px;
  line-height: 18px;
  font-size: 14px;
}

.active {
  box-shadow: 0px 8px 8px #f69038;
}

.container__start{
  display: none;
  justify-content: center;
  align-items: center;
}

.card__number,
.card__title,
.card__badge {
  margin: 12px 0;
}
```

```
192
193 .card__number{
194     color: white;
195     font-size: 1rem;
196     font-weight: bold;
197 }
198
199 #start {
200   display: none;
201 }
202
203 @keyframes bounce {
204   20% {
205     transform: translateY(-6px);
206   }
207   40% {
208     transform: translateY(0px);
209   }
210
211   80% {
212     transform: translateY(-2px);
213   }
214   100% {
215     transform: translateY(0);
216   }
217 }
218
219 @media screen and (max-width: 1200px) {
220   .card {
221     width: 236px;
222   }
223 }
224
225 @media screen and (max-width: 520px) {
226   .card {
227     width: 350px;
228   }
229
230 }
231
```

```
@media screen and (max-width: 460px) {
  .btn__compare, .btn__start {
    margin: 16px 0;
  }
  .container__options {
    justify-content: center;
    flex-direction: column;
  }
}
```

B.8 *reset.css*

```
/* http://meyerweb.com/eric/tools/css/reset/
v2.0 | 20110126
License: none (public domain)
*/

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
  margin: 0;
  padding: 0;
  border: 0;
  font-size: 100%;
  font: inherit;
  vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
```

```
footer, header, hgroup, menu, nav, section {
  display: block;
}
body {
  line-height: 1;
}
ul {
  list-style: none;
}
blockquote, q {
  quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
  content: '';
  content: none;
}
table {
  border-collapse: collapse;
  border-spacing: 0;
}
```