UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA
COLEGIADO DO CURSO DE ENGENHARIA DA COMPUTAÇÃO

Beatriz Ogioni Sessa

UMA FERRAMENTA BASEADA EM ONTOLOGIA PARA APOIAR ASPECTOS DE GERÊNCIA DE CONHECIMENTO NO DESIGN DE IHC

Projeto de Graduação apresentado ao Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia da Computação.

Orientadora: Monalessa Perini Barcellos
Coorientador: Murillo Vasconcelos Henriques Bittencourt Castro

VITÓRIA
2021

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA
COLEGIADO DO CURSO DE ENGENHARIA DA COMPUTAÇÃO

Beatriz Ogioni Sessa

UMA FERRAMENTA BASEADA EM ONTOLOGIA PARA APOIAR ASPECTOS DE GERÊNCIA DE CONHECIMENTO NO DESIGN DE IHC

COMISSÃO EXAMINADORA

_____

Prof. Monalessa Perini Barcellos, D. Sc.

_____

Prof. Davidson Cury, D. Sc.

_____

Prof. Simone Dornellas Costa, M. Sc.

Vitória, 20 de maio de 2021

A Mônica e Jezuino, que se mantiveram firmes
e me mantiveram firme no momento
mais difícil das nossas vidas.

AGRADECIMENTOS

Nunca terei palavras suficientes para agradecer meus pais, Mônica e Jezuino, pela força indescritível, amor e dedicação. Essa conquista é nossa. Aos meus avós maternos, por serem os maiores entusiastas de absolutamente todos os meus projetos. Aos meus avós paternos, pela fonte inesgotável de carinho e serenidade.

A vida me deu o privilégio de, além de um irmão de sangue, me dar vários outros de consideração. Agradeço ao Victor, que, da sua maneira, sempre foi um dos maiores apoiadores das minhas escolhas. A Jady e Leonardo, que mesmo sendo primos, me adotaram como irmã caçula e me acompanham em todas as fases da minha vida. A Marina, que mesmo de longe, é uma constante. Agradeço imensamente por todas as horas no telefone, escutando meus desabafos e abraçando todas as minhas dores como se fossem suas. Morro de orgulho por poder chamar vocês de irmãos.

A toda a minha família, por todo amor incondicional e presença constante. Em especial, aos meus primos, Tiago, Amanda, Carolina, Luiza, Arthur, Hugo e João, que sempre me fizerem rir independente das circunstâncias. É um privilégio ter crescido com vocês e ser parte dessa família.

Ao longo da graduação pude fazer mais amizades que um dia me permiti sonhar. Agradeço ao Gabriel Rezende e ao Lucio Sandrini, por serem os melhores parceiros de laboratório e grupo de trabalho. Não teria conseguido sem vocês dois. A Nicole, Úrsula e Isabel, que muitas vezes dividiram a experiência de serem as únicas meninas em sala de aula junto comigo. Ao Guilherme, que colou em mim desde o primeiro dia de aula e nunca me deixou esquecer o que eu realmente queria. Ao Luiz Antônio, que foi o melhor veterano e monitor de programação do mundo. Ao Felipe, que escutou todos os meus desabafos e questionamentos e me apoiou imensamente. Ao Gabriel Giorisatto, Douglas, Luiz Otavio, Gustavo, Gilmarllen e Lucas, por terem sido fonte de inspiração durante toda minha graduação. A todos os outros incontáveis amigos que me acompanharam durante esses anos, deixando os dias mais leves e divertidos. A Maria Clara, pela amizade que já dura toda nossa vida. A todo o time do PET Engenharia da Computação, por tantas oportunidades incríveis.

Ao meu coorientador, Murillo, por todo apoio, paciência e dedicação ao logo desse trabalho. A minha orientadora Monalessa, pela oportunidade e incentivo. A todo o departamento de informática, por tantas aulas incríveis e pelo incentivo a busca constante por conhecimento, em especial as professoras Roberta e Patrícia, por todo o apoio.

RESUMO

Com a democratização dos computadores e *smartphones*, é cada vez maior o investimento feito por corporações nos sistemas computacionais interativos. Visando melhorar cada vez mais a usabilidade desses sistemas, são levados em consideração aspectos de design de interação humano-computador (IHC). Como IHC é uma área multidisciplinar, equipes envolvidas no design de IHC costumam ser compostas por profissionais de diferentes especialidades, cada qual com seu próprio corpo de conhecimento e vocabulário técnico. Com isso, a conceituação sobre design de IHC e sobre o produto sendo desenvolvido pode se tornar conflitante entre os diferentes profissionais (i.e., não há consenso com relação ao significado dos conceitos), dificultando assim a comunicação e a transferência de conhecimento. Diante disso, a combinação entre ontologias e soluções de gerência de conhecimento (GC) pode ser útil, apoiando a captura e organização de conhecimento de design de IHC e provendo mecanismos que possibilitam a representação, armazenamento, recuperação, utilização e avaliação desse conhecimento.

Nesse contexto, a ontologia de referência HCIDO (*Human-Computer Interaction Design Ontology*) foi proposta em (CASTRO, 2021), com o intuito de prover uma conceituação bem fundamentada e consensual sobre design de IHC e servir de apoio para soluções de GC no design de IHC. Neste trabalho, HCIDO foi utilizada no desenvolvimento de KTID (*Knowledge Tool for Interaction Design*), uma ferramenta de apoio a aspectos de GC no design de IHC. HCIDO foi utilizada na modelagem conceitual de KTID como um recurso de conhecimento sobre o domínio de design de IHC. Uma versão inicial de KTID foi projetada, desenvolvida e disponibilizada para uso, incluindo funcionalidades que apoiam representação, armazenamento, recuperação e avaliação de conhecimento no design de IHC.

Palavras-chave: Design de IHC, Ontologias, Gerência de Conhecimento

LISTA DE FIGURAS

LISTA DE TABELAS

SUMARIO

Capítulo 1
Introdução

Este capítulo apresenta uma breve introdução ao tema do trabalho, seus objetivos, histórico do desenvolvimento e a organização deste documento.

1.1 Introdução

O avanço da internet e democratização dos computadores causou o surgimento de inúmeros sistemas computacionais interativos, que consistem em uma combinação de hardware e software que recebe comandos de entrada e comunica respostas como saída para os usuários (ISO, 2019). Com isso, um número crescente de esforços de design e engenharia se voltaram para a experiência do usuário e para a ampliação do uso de recursos desses sistemas, visando promover estudos e práticas a respeito da usualidade (CARROLL, 2014). Usabilidade está relacionada à eficiência e satisfação sob o ponto de vista do usuário, sendo considerada um dos aspectos chave para o sucesso de um sistema interativo. Para atingir um alto nível de usabilidade, é importante levar em conta aspectos do design de interação humano-computador (IHC) no processo de desenvolvimento do sistema interativo (CARROLL, 2014).

O design de IHC está relacionado também a outros aspectos da interação entre usuários e sistemas computacionais interativos, envolvendo conhecimento de diferentes campos, como ergonomia, ciência cognitiva, experiência do usuário, fatores humanos e vários outros (CARROLL, 2014; SUTCLIFFE, 2014). Devido a essa diversidade de conhecimento presente no desenvolvimento de sistemas interativos, a equipe envolvida nesse processo é frequentemente multidisciplinar, incluindo pessoas de diferentes especializações, com suas próprias linguagens técnicas, jargões, termos e experiências. Com isso, até mesmo a conceituação sobre o produto sendo desenvolvido pode ser conflitante entre os envolvidos, o que dificulta a comunicação e a transferência de conhecimento entre as partes interessadas (ROGERS, 2011; CARROLL, 2014).

Desenvolver um sistema de software é uma tarefa intensiva em conhecimento, que tem sido apoiada por princípios e práticas da gerência de conhecimento (GC) para o suporte, captura, utilização e transferência de conhecimento (RUS; LINDVALL, 2002; VALASKI, 2012). Tais princípios podem ser úteis também para abordar os desafios específicos do design de IHC de sistemas interativos. Por exemplo, pode-se desenvolver uma estratégia para armazenar e recuperar informações sobre requisitos, componentes de design e padrões de

design utilizados na construção de uma solução de design. Como resultado, a equipe pode aprender com as experiências anteriores e compartilhar conhecimento comum a respeito do design de IHC do sistema, contribuindo assim para a produção de melhores produtos e tornando o processo de desenvolvimento mais eficiente.

IHC é uma área muito ampla e, conforme fica mais madura, novos termos vão sendo propostos, assim como novos significados vão sendo associados a termos previamente existentes. Como consequência, não é trivial ter uma conceituação comum sobre IHC e design de IHC, causando problemas de imprecisão na definição e ambiguidade na interpretação de conceitos compartilhados. Ontologias podem ser úteis para apoiar a captura e organização de conhecimento no design de IHC, uma vez que podem ser utilizadas para estabelecer uma conceituação formal e comum de um domínio particular de interesse. Uma ontologia é uma maneira formal e explícita de especificar uma conceituação compartilhada (STUDER, 1998). No domínio de IHC, ontologias têm sido aplicadas para representação de conhecimento, apoio ao design e avaliação de IHC, adaptação de interfaces e anotação semântica, entre outros (COSTA et al., 2020; COSTA, 2021). Nesse contexto, foi desenvolvida em um trabalho de mestrado a ontologia de referência HCIDO (*Human-Computer Interaction Design Ontology*) (CASTRO, 2021), com o intuito de prover uma conceituação bem fundamentada e consensual sobre design de IHC.

Além disso, sendo ferramentas adequadas para a representação de conhecimento, ontologias têm sido usadas para apoiar soluções de GC (ex.: sistemas de gerência de conhecimento, sistemas baseados em conhecimento ou até soluções não automatizadas). Essas soluções podem utilizar do suporte de ontologias para prover acesso a conhecimento, otimização e recuperação do conhecimento e apoio a mecanismos de comunicação que auxiliam a troca de conhecimento fornecido pela ontologia (VARMA, 2007). Com isso, o uso de ontologias combinado com soluções de GC no design de IHC poderia aprimorar a utilização e facilitar o raciocínio sobre conhecimento de IHC já existente. Nesse sentido, HCIDO foi proposta como um recurso para dar apoio a soluções de GC no design de IHC (CASTRO, 2021). Como forma de demonstrar uma prova de conceito da aplicação de HCIDO para soluções de GC, a ontologia foi usada como base para a modelagem conceitual de KTID (*Knowledge Tool for Interaction Design*), uma ferramenta de apoio a aspectos de GC no design de IHC, cujo desenvolvimento é tratado neste trabalho.

1.2 Objetivos

Este trabalho tem como objetivo geral *desenvolver uma ferramenta baseada no modelo conceitual de HCIDO (CASTRO, 2021) para apoiar aspectos de gerência de conhecimento no design de IHC.* A ferramenta deve proporcionar o compartilhamento de conhecimento sobre escolhas de design feitas em soluções de design propostas para sistemas interativos computacionais, implementando funcionalidades que apoiem atividades de representação, armazenamento, recuperação e avaliação desse conhecimento. Dessa forma, o conhecimento torna-se explícito para outras pessoas envolvidas no desenvolvimento do sistema interativo (ex.: designers, desenvolvedores e gerentes de projeto), melhorando a comunicação entre elas e permitindo que o conhecimento possa ser reutilizado em iniciativas futuras. Esse objetivo geral pode ser detalhado nos seguintes objetivos específicos:

i. Identificar itens de conhecimento relevantes no design de IHC para serem tratados na ferramenta;
ii. Elaborar a modelagem conceitual da ferramenta a partir do modelo conceitual de HCIDO (CASTRO, 2021), uma ontologia de referência sobre design de IHC;
iii. Definir e projetar uma arquitetura para a ferramenta, levando em considerações padrões, modelos e paradigmas utilizados no projeto de sistemas de software;
iv. Implementar, testar e disponibilizar para uso uma versão inicial da ferramenta.

1.3 Histórico de Desenvolvimento do Trabalho

Este trabalho foi conduzido de acordo com as seguintes atividades:

i. *Revisão da Literatura:* O trabalho teve início com uma revisão bibliográfica sobre design de IHC, gerência de conhecimento, ontologias e redes de ontologias, na qual foram lidos materiais (livros, teses, dissertações e artigos científicos) pertinentes ao assunto.
ii. *Estudo de Tecnologias:* Nesta etapa ocorreu o estudo de tecnologias relevantes para o desenvolvimento da ferramenta, destacando-se: o *framework Vue.js*, que utiliza as linguagens *JavaScript*, HTML e CSS, para o lado do cliente da aplicação, e o *framework Laravel*, que utiliza PHP como linguagem, para o lado do servidor. Também ocorreu o estudo dos padrões MVVM (*Model-View-ViewModel*), utilizado no *Vue.js,* e MVC (*Model-View-Controller*), utilizado no *Laravel.*

iii. *Levantamento e Análise de Requisitos*: Consistiu na identificação e análise dos requisitos da ferramenta a partir da conceituação de HCIDO (CASTRO, 2021), incluindo a elaboração de diagramas da UML (*Unified Modeling Language)*, tais como diagrama de casos de uso e diagrama de classes. Após isso, utilizando-se o método prototípico, foram desenvolvidos *wireframes* para esboçar a o design de interface com o usuário da ferramenta, com as definições de design de interface.

iv. *Design, Implementação e Testes:* Nesta etapa ocorreu o projeto e implementação de componentes, páginas do lado do cliente e integração com a lógica de negócio do lado do servidor. Foram feitos testes ao longo do desenvolvimento do projeto e no momento de conclusão das funcionalidades.

v. *Escrita da Monografia:* Consistiu na escrita desta monografia.

1.4 Organização do Texto:

Neste capítulo inicial, está contida a introdução do trabalho, assim como seus objetivos e seu histórico de desenvolvimento. Além deste capítulo, a monografia possui outros três, a saber:

Capítulo 2 – Fundamentação Teórica: Apresenta uma breve fundamentação teórica sobre design de IHC, gerência de conhecimento e ontologias, com destaque para HCIDO. Também apresenta as tecnologias que foram utilizadas no desenvolvimento da ferramenta.

Capítulo 3 – Desenvolvimento da ferramenta KTID: Apresenta os requisitos da ferramenta, a descrição de seu contexto e propósito de uso, seus casos de uso, modelos de classe, arquitetura e detalhes de componentes da arquitetura. Também descreve as funcionalidades implementadas da ferramenta e apresenta um fluxo de sua utilização.

Capítulo 4 – Considerações finais: Apresenta as considerações finais do trabalho, incluindo algumas dificuldades encontradas, limitações da ferramenta desenvolvida e experiências adquiridas. Além disso, são identificados alguns trabalhos futuros.

Capítulo 2
Fundamentação Teórica

Este capítulo apresenta os principais aspectos teóricos que fundamentam este trabalho. Ele está organizado em 5 seções, a saber: Design de IHC (Seção 2.1), Gerência de Conhecimento (Seção 2.2), Ontologias (Seção 2.3), Tecnologias envolvidas nesse trabalho (Seção 2.4) e Considerações finais do capítulo (Seção 2.5).

2.1 Design de IHC

Interação Humano Computador (IHC) pode ser definida como uma disciplina responsável pela análise, design, implementação e avaliação de sistemas computacionais interativos (ROGERS; SHARP; PREECE, 2011). Como dito no capítulo anterior, um sistema computacional interativo (também referido como "sistema interativo" neste trabalho), por sua vez, combina software e hardware que recebe uma entrada de um usuário e comunica uma resposta como saída para ele (ISO, 2019). Dix *et al.* (2003) consideram o fenômeno da interação humano-computador como um processo de comunicação que ocorre durante o uso de um sistema de computação interativo, envolvendo ações do usuário na interface do sistema e as respostas desse sistema ao usuário, exibidas por meio de uma interface.

Design de IHC tem como foco o projeto de sistemas interativos para apoiar usuários a atingirem seus objetivos por meio de interações com o sistema (SUTCLIFFE, 2014). Para isso, são levados em conta diversos aspectos, como usabilidade, experiência do usuário, acessibilidade e comunicabilidade. *Usabilidade* é a extensão na qual um sistema, produto ou serviço pode ser usado por determinados usuários para atingir objetivos com eficiência, efetividade e satisfação em um contexto de uso específico (ISO 9241-210, 2010). Para isso, observa-se o esforço e a facilidade do usuário durante a interação, considerando suas habilidades cognitivas, perceptivas e motoras. A definição de *experiência do usuário* está relacionada às suas emoções e sentimentos ao usar um produto, sendo essencial para o design de interação por levar em consideração como o produto se comporta ao ser usado pelas pessoas no mundo real (ROGERS, 2011). A *acessibilidade*, por sua vez, faz referência à remoção de barreiras que impedem a interação com a interface e o acesso a ela. Por fim, a *comunicabilidade* se concentra na capacidade de a interface conseguir comunicar um design lógico ao usuário (DE SOUZA, 2015).

O design de IHC centrado no usuário, é dito *User-Centered Design* (UCD) (CHAMMAS, 2015). UCD coloca as necessidades, capacidades e comportamentos humanos em primeiro plano, fazendo com que o design do sistema os acomode. Seus princípios básicos são: foco no usuário (suas características, necessidades e objetivos), métricas observáveis (desempenho e reações do usuário) e design iterativo (permitindo repetição quantas vezes for necessário). O termo *Human-Centered Design* (HCD) tem sido adotado no lugar de UCD para enfatizar o impacto sobre todas as partes interessadas e não apenas sobre aqueles considerados usuários (ISO 9241-210, 2019).

Em linhas gerais, o processo de UCD envolve: *compreensão e especificação do contexto de uso*, que tem como meta estudar os usuários do produto e os usos pretendidos; *especificação de requisitos*, que visa identificar as necessidades do usuário e de requisitos funcionais e não funcionais para o produto; *produção de soluções de design*, que visa à obtenção da melhor experiência do usuário e inclui a produção de artefatos, como protótipos e esboços, que serão utilizados no futuro como base para o desenvolvimento do sistema; e *avaliação*, quando são avaliados os resultados produzidos nas atividades anteriores (ISO 9241-210, 2019).

Por ser um processo intensivo em conhecimento, design de IHC pode obter grandes vantagens da utilização de soluções de GC, através da aplicação de mecanismos efetivos de criação colaborativa e de compartilhamento de conhecimento de design (ex.: informações sobre os usuários, sistemas, propósitos de utilização e contextos de uso), apoiando o reúso de conhecimento empregado em inciativas anteriores e melhorando a comunicação entre os envolvidos.

2.2 Gerência de Conhecimento

Conhecimento é uma especialidade humana gravada na mente das pessoas, adquirida através da experiência e da interação com o ambiente (SCHNEIDER, 2009). Polanyi (1966) classifica conhecimento em dois tipos diferentes, o tácito e o explícito. O *conhecimento tácito* representa o conhecimento que é subjetivo e não documentável, que existe apenas na mente das pessoas, envolvendo fatores como experiências prévias, crenças e intuição. Já o *conhecimento explícito* representa todo conhecimento que pode ser documentado e acessado por outras pessoas, sendo facilmente transmissível e compartilhável, por meio de fórmulas cientificas, princípios gerais e outras maneiras. Historicamente, o conhecimento organizacional não era documentado, sendo expresso através de habilidades, experiências e conhecimento de seus profissionais, tipicamente conhecido como conhecimento tácito (RUS, 2002), o que tornava seu acesso limitado e difícil (O'LEARY, 1998).

A Gerência de Conhecimento (GC) visa transformar o conhecimento tácito e individual em conhecimento explícito e compartilhado. Ao transformar o conhecimento individual em conhecimento organizacional, a GC promove a propagação de conhecimento e aprendizado, tornando o conhecimento acessível e reutilizável para toda a organização (SCHNEIDER, 2009; RUS, 2002; O'LEARY, 1998). Tal conhecimento ajuda empresas de software e tecnologia a reagirem mais rapidamente e da melhor maneira possível, tendo respostas mais precisas e aumentando assim a qualidade do software e a satisfação dos clientes (SCHNEIDER, 2009).

Quando uma organização implementa GC, as experiências e conhecimentos são armazenados, avaliados, preservados, projetados e sistematicamente propagados para resolver problemas (SCHNEIDER, 2009). Portanto, a GC aborda o conhecimento no seu círculo evolutivo, que consiste na sua criação, captura, transformação, acesso e aplicação.

No contexto do processo de desenvolvimento de sistemas de software, a GC atua gerenciando conhecimento de forma explícita e sistemática, lidando com a sua aquisição, armazenagem, organização, evolução, ativação e utilização. Ela vem sendo aplicada no contexto de desenvolvimento de software para suporte à gestão de documentos e competências, identificação de experts, reutilização de software, suporte ao aprendizado e memória de produtos e projetos (RUS, 2002).

Uma investigação do estado da arte e estado da prática sobre a utilização de GC no design de IHC apontou que GC tem sido usada no design de IHC principalmente para melhorar a qualidade do produto e reduzir o esforço e o tempo gastos nas atividades de design (CASTRO, 2021). Outros resultados dessa investigação também indicaram que a falta de uma conceituação comum sobre design de IHC leva a problemas de comunicação entre os diferentes atores envolvidos nesse processo, e que existe uma lacuna entre teoria e prática. Assim, torna-se necessário levar soluções de GC para ambientes mais práticos de design de IHC, já que, apesar de profissionais de design de IHC estarem habituados a gerenciar conhecimento, eles costumam utilizar soluções mais simples e práticas.

Considerando esse cenário, ontologias podem ser úteis para prover uma fundamentação consistente e uma conceituação consensual do domínio do design de IHC, podendo ser utilizadas para apoiar soluções de GC nesse contexto.

2.3 Ontologias

Uma ontologia é uma especificação formal e explícita de uma conceituação compartilhada (STUDER, 1998). Ontologias podem ser organizadas em três camadas arquiteturais (SCHERP, 2011) sendo elas: *(i) ontologias de fundamentação,* que focam em modelar conceitos e relações básicos e gerais que compõem o mundo (como objetos e eventos) e, por serem genéricas, têm altíssimo poder de reúso em diferentes cenários de modelagem; *(ii) ontologias de núcleo*, que proporcionam o refinamento das ontologias de fundamentação*,* com a adição de conceitos detalhados e relações sobre uma área específica (como um serviço, processo, estrutura organizacional), mas ainda se estendendo por vários domínios; e por fim, *(iii) ontologias de domínio* incluem ontologias altamente especializadas, que têm a melhor descrição possível do conhecimento de uma determinada área do domínio na realidade. Uma ontologia descrevendo o fenômeno do design de IHC de sistemas interativos é um exemplo de ontologia de domínio, que pode fazer uso de ontologias das duas camadas anteriores, especializando seus conceitos.

Outra importante distinção é entre ontologias como modelos conceituais, chamadas de *ontologias de referência*, e ontologias como artefatos computacionais, chamadas de *ontologias operacionais* (GUIZZARDI, 2007). Uma *ontologia de referência* é construída com o objetivo de fazer a melhor descrição possível do domínio em questão, representando o modelo que melhor descreva a situação, sem se preocupar com suas propriedades computacionais. Já uma *ontologia operacional*, por sua vez, é feita com foco em garantir a realização de funções computacionais e, com isso, devem ser descritas em linguagens entendíveis por máquinas.

Para um domínio mais complexo, a representação de conhecimento como uma ontologia única resulta em um grande monolito, que se torna difícil de manipular, usar e manter. Em contrapartida, representar cada subdomínio isoladamente é uma tarefa custosa, com alta fragmentação, onde as mesmas complexidades da alternativa anterior acabam ocorrendo (RUY, 2016). Em vista desses fatores, uma melhor solução pode ser a construção de uma rede de ontologias, ou seja, uma coleção de ontologias relacionadas entre si por meio de várias ligações (SUÁREZ-FIGUEROA, 2012).

A ferramenta desenvolvida neste trabalho foi proposta com base no conhecimento do domínio de design de IHC provido pela ontologia de referência *Human-Computer Interaction Design Ontology* (HCIDO) (CASTRO, 2021). HCIDO é uma ontologia de domínio de HCI-ON (*Human-Computer Interction Ontology Network*) (COSTA *et al.*, 2020), uma rede de ontologias que aborda vários aspectos de IHC (ex.: o fenômeno de interação humano-computador,

avaliação de IHC, tipos e elementos de interface com o usuário, entre outros), e reúsa conceitos de SEON (*Software Engineering Ontology Network*) (RUY *et al.*, 2016), uma rede de ontologias que descreve vários subdomínios de Engenharia de Software (ex.: requisitos de software, implementação, testes, medição de software, etc.). Tanto as ontologias de HCI-ON quanto as de SEON estão fundamentadas em UFO (*Unified Foundational Ontology*) (GUIZZARDI, 2005), uma ontologia de fundamentação que aborda vários aspectos essenciais para a modelagem conceitual do domínio de design de IHC, como agentes, objetos e propriedades mentais.

A Figura 1 apresenta o posicionamento de HCIDO na arquitetura de HCI-ON, assim como as relações de HCIDO com outras ontologias das redes. Na Figura, cada um dos círculos (nó de rede) representa uma ontologia de núcleo ou domínio de HCI-ON ou SEON. Círculos pontilhados representam ontologias de HCI-ON em desenvolvimento. Setas vermelhas representam relacionamentos de dependência de HCI-ON para SEON. As dependências de ontologias de HCI-ON para ontologias de núcleo de SEON são denotadas por setas sólidas em vermelho, enquanto as dependências para ontologias de domínio de SEON são denotadas por setas pontilhadas em vermelho.

A seção a seguir apresenta HCIDO e a descrição de seus conceitos que foram utilizados no desenvolvimento deste trabalho.

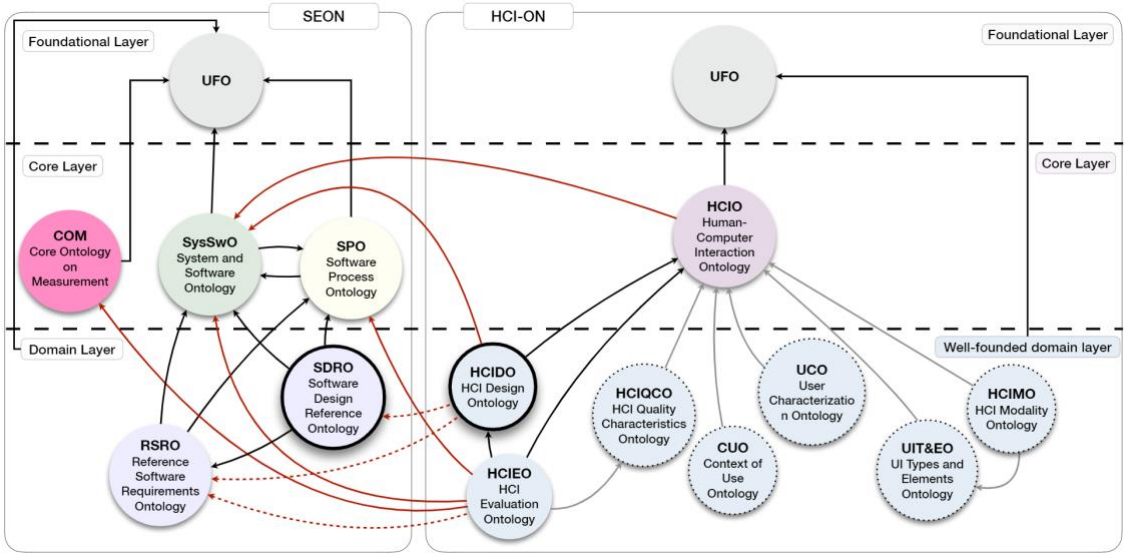

Figura 1 - Posicionamento de HCIDO com relação às arquiteturas de HCI-ON e SEON (CASTRO, 2021)

2.3.1 Human-Computer Interaction Design Ontology (HCIDO)

A *Human-Computer Interaction Design Ontology* (HCIDO) (CASTRO, 2021) foi proposta para prover uma conceituação formal bem fundamentada sobre design de IHC. Em sua versão atual, HCIDO aborda o design de software de sistemas interativos, não abordando aspectos relacionados a hardware. Ela aborda uma interseção de conhecimentos entre os domínios de IHC e Engenharia de Software, conectando conceitos centrais de IHC providos por HCIO – *Human-Computer Interaction Ontology* (uma ontologia de núcleo de HCI-ON) com conceitos de design de software providos por SDRO – *Software Design Reference Ontology* (uma ontologia de domínio de SEON) (CASTRO, 2021). SDRO descreve os elementos mentais e físicos envolvidos no design de sistemas de software e as relações entre eles. HCIDO reutiliza esta distinção de SDRO e caracteriza aspectos específicos relacionados à especificação de design e ao objeto de design, que possuem diferenças no contexto de design de IHC em relação ao domínio de design de software. Com isso, HCIDO é decomposta em duas subontolgias: a subontologia de Especificação de Design (*Design Specification*) e a subontologia de Objeto de Design (*Design Object*). A arquitetura de HCIDO é apresentada na Figura 2.

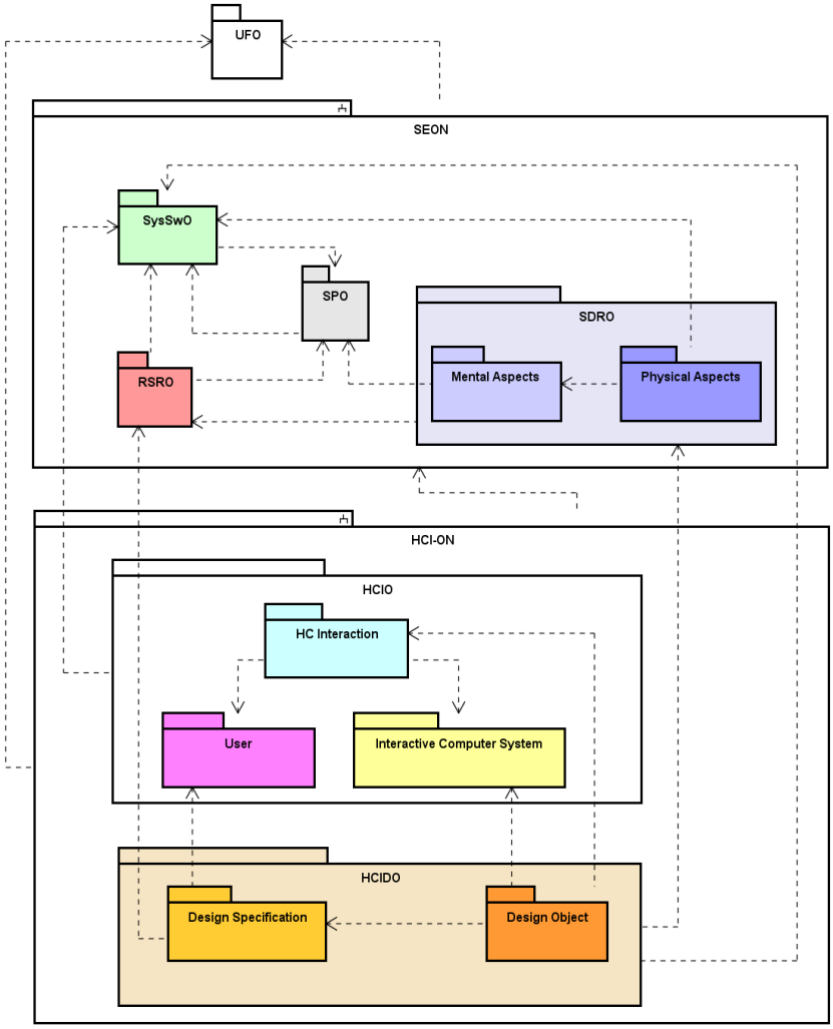

Figura 2 - Arquitetura de HCIDO (CASTRO, 2021)

A subontologia de Especificação de Design da HCIDO é apresentada na Figura 3 e a subontologia de Objeto de Design é apresentada na Figura 4. Nas figuras, as linhas duplas, pontilhadas em vermelho separam as camadas das arquiteturas que correspondem a SEON e a HCI-ON, de acordo com a classificação proposta por Sherp *et al.* (2011). As linhas pretas pontilhadas separam conceitos de diferentes ontologias na mesma camada e o nome de cada uma das ontologias está indicada na figura próximo à borda do lado direto. Conceitos de HCIDO são apresentados em diferentes tons de laranja, de acordo com a subontologia correspondente: laranja claro é usado para a subontologia Especificação de Design e laranja escuro é usado para a subontologia Objeto de Design. Os conceitos que foram diretamente considerados para o desenvolvimento do modelo conceitual de KTID estão destacados com uma borda azul. Após a figura, é apresentada uma breve descrição desses conceitos. Uma descrição completa e detalhada dos modelos pode ser encontrada em (CASTRO, 2021).

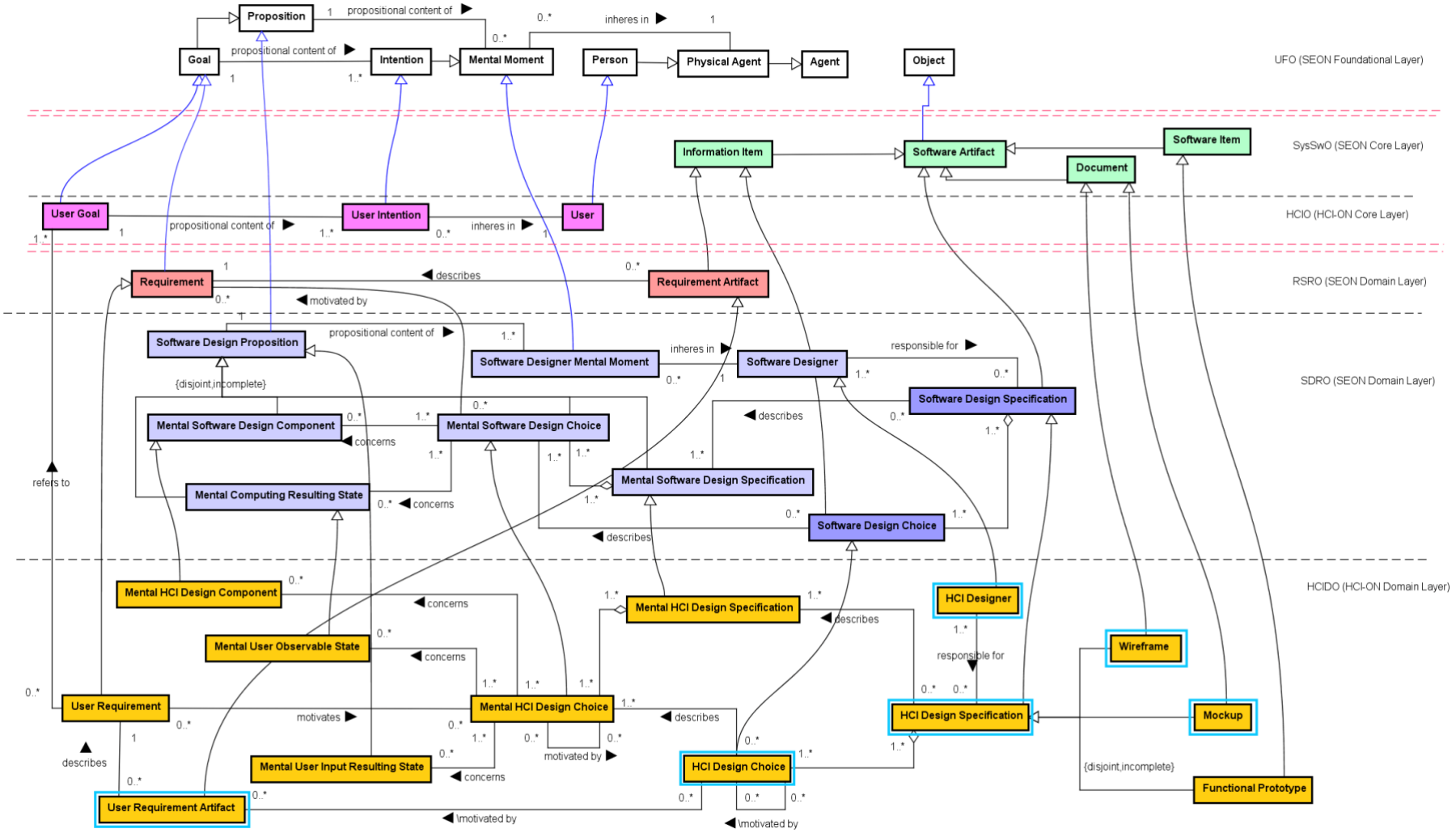

Figura 3 – Modelo conceitual da subontologia Especificação de Design de HCIDO (adaptado de (CASTRO, 2021))

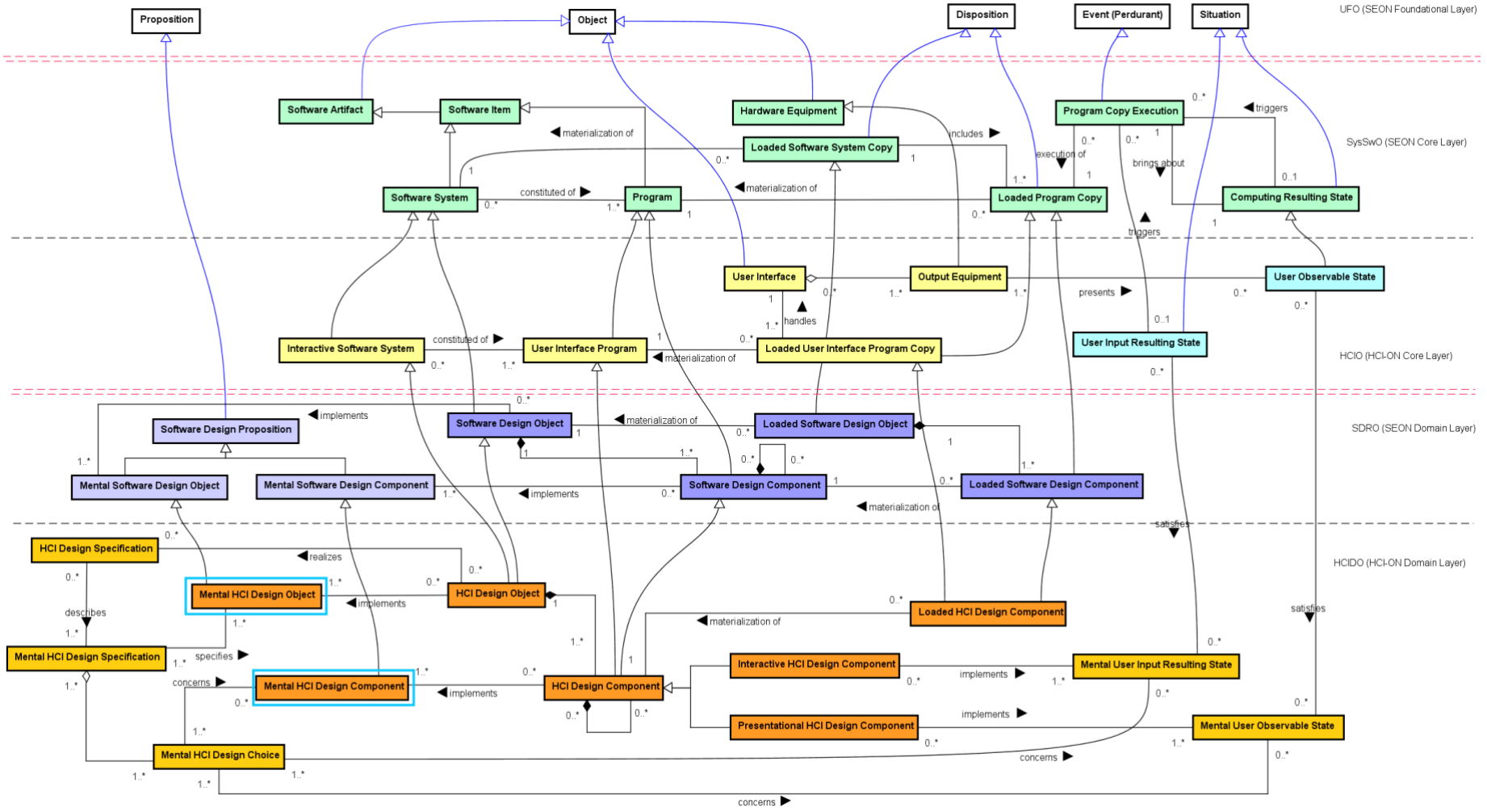

Figura 4 - Modelo conceitual da subontologia Objeto de Design de HCIDO (adaptado de (CASTRO, 2021))

Um **HCI Designer** é um agente responsável por utilizar suas habilidades para contribuir com a criação de uma especificação que contenha escolhas de design sobre aspectos da interação humano-computador de um sistema de software interativo a ser desenvolvido. As ideias na mente de um **HCI Designer** a respeito de descrições detalhadas de aspectos estruturais e comportamentais de interação humano-computador do sistema interativo são representadas na ontologia pelo conceito **Mental HCI Design Choice**. Os registros físicos[1] dessas ideias, criados para propósitos de comunicação e avaliação, são itens de informação representados pelo conceito **HCI Design Choice**. **Mental HCI Design Choices** podem ser motivadas por outras **Mental HCI Design Choices** ou por **User Requirements**, que são requisitos de usuário que o sistema interativo precisa atender para que usuários atinjam seus objetivos com a utilização do sistema. Requisitos de usuário são descritos por itens de informação denominados **User Requirement Artifacts**, que correspondem, portanto, ao registro físico de **User Requirements**.

Uma **Mental HCI Design Specification** representa o conjunto de todas **Mental HCI Design Choices** que estão na mente de um designer a respeito de um determinado sistema sendo desenvolvido. **Mental HCI Design Specifications** podem ser descritas por artefatos denominados **HCI Design Specifications**, que podem ser documentos, modelos ou até trechos de código. Três tipos de **HCI Design Specification** são identificados em HCIDO: **Wireframes**, **Mockups** e **Functional Prototypes**. Neste trabalho, apenas os dois primeiros foram utilizados. **Wireframes** representam com baixa fidelidade a estrutura básica da interface a ser desenvolvida para o sistema interativo. Um **Mockup**, por sua vez, representa o sistema a ser desenvolvido com alta fidelidade, sendo visualmente o mais próximo possível do resultado final das telas a serem implementadas no sistema interativo.

Ao projetar um sistema interativo, designers costumam se referir ao sistema ou a suas partes mesmo que eles ainda não existam na realidade. Entretanto, na mente deles existe uma imagem do que virá a se concretizar como o objeto de design e as partes que o compõem. Para representar esse tipo de situação, HCIDO propõe os conceitos **Mental HCI Design Object** e **Mental HCI Design Component**. Eles representam o que designers imaginam sendo o futuro sistema interativo e seus componentes, respectivamente. Após a implementação do

[1] Aqui, o termo *físico* significa que a informação está explícita de alguma forma (por exemplo, em um documento impresso ou em um modelo armazenado na memória do computador).

sistema interativo, o sistema é considerado um **HCI Design Object** caso implemente exatamente o que o designer imaginou e especificou. Da mesma forma, **HCI Design Components** são partes menores do sistema (ex.: programas) que implementam os componentes especificados pelo designer.

HCIDO foi desenvolvida visando prover suporte a soluções de GC no design de IHC. Sua conceituação permite entender melhor o processo de criação e realização de escolhas do designer de IHC, evidenciando como elementos mentais se relacionam com os artefatos físicos que os descrevem. Com isso, HCIDO pode apoiar o desenvolvimento de soluções de GC que auxiliam no processo de transformação de conhecimento tácito em conhecimento explícito no design de IHC.

2.4 Tecnologias Envolvidas neste Trabalho

Nesta seção são apresentadas breves descrições das tecnologias utilizadas neste trabalho. A ferramenta foi desenvolvida como uma aplicação baseada na Web (também conhecida como *WebApp* (PRESSMAN; LOWE, 2009), seguindo o modelo cliente-servidor, onde a parte relativa ao cliente é chamada também de *front-end* e a parte relativa ao servidor é chamada de *back-end*. Uma prática muito comum na construção de *WebApps* é a utilização de *frameworks*. Em Engenharia de Software, um *framework* é uma estrutura que fornece códigos reutilizáveis de forma abstrata, que podem nos mais diversos projetos (RIEHLE, 2000). O objetivo ao usá-los é não despender tempo fazendo atividades repetitivas e que são comuns a vários sistemas (ex.: controle de autenticação e autorização para funcionalidades). Neste trabalho, foram utilizados *frameworks* tanto no lado do servidor quanto no lado do cliente. Os *frameworks* utilizados são apresentados nas seções a seguir.

2.4.1. Back-end: *frameworks* MVC e Laravel

O *back-end* da aplicação foi desenvolvido baseado no padrão de arquitetura MVC, abreviação de *Model-View-Controller* (GAMMA, HELM, JOHNSON, VLISSIDES, 1994), constituído por três camadas (representadas na Figura 5): a camada de Modelo (*Model*), responsável pela lógica de negócio da aplicação; a camada de Visão (*View*), que é a camada de interação com o usuário; e a camada de Controle (*Controller*), que faz a comunicação entre as camadas citadas anteriores.

Figura 5 - Diagrama representando o padrão MVC

Para a implementação do padrão MVC na ferramenta, foi utilizado o *framework open source Laravel*[2], que é construído com a linguagem de programação PHP[3] e é massivamente utilizado para desenvolvimento de servidores para sistemas web, como é o caso da aplicação desenvolvida neste trabalho. Essa escolha se deu devido ao fato desse *framework* ter uma documentação vasta e de fácil compreensão, diminuindo assim a curva de aprendizado para sua utilização. Além disso, a aplicação do padrão de arquitetura MVC de forma clara no *framework* auxilia no desenvolvimento e manutenibilidade do código. *Laravel* tem ainda várias funcionalidades que visam abstrair dificuldades para o programador e manter a consistência do projeto, como é o caso das *Migrations,* que garantem que o banco de dados será criado com os mesmos parâmetros e populado com os mesmos valores, sempre na mesma ordem.

A implementação do *back-end* com o *framework Laravel* neste trabalho utilizou as seguintes linguagens:

- **PHP: *Hypertext Preprocessor* (PHP):** Linguagem de programação interpretada livre, que recomendada para o desenvolvimento de aplicações para os servidores de sistemas web.
- **MYSQL[4]:** Linguagem de gerenciamento de banco de dados, que utiliza a linguagem de consulta estruturada, o SQL. Essa escolha se deu por esse ser um dos sistemas de gerenciamento de banco de dados mais populares, com vasta documentação.

No Capítulo 3, será detalhada a implementação do padrão MVC a partir da utilização do *Laravel.*

[2] **https://laravel.com**
[3] **https://www.php.net**
[4] **https://www.mysql.com**

2.4.2. Front-end: *frameworks* MVVM e Vue.js

O *front-end* da aplicação foi desenvolvido a partir do padrão MVVM (Model, View, ViewModel). Esse padrão foi apresentado pela primeira vez por Jonh Gossman (Smith, 2009), sendo ele uma variação do padrão PM (*Presentation Model*). É importante notar que ambos MVVM e PM são derivados do padrão MVC. O padrão MVVM consiste na divisão do projeto em três camadas de responsabilidades, como pode ser observado na Figura 6. A *ViewModel* é a camada responsável por comunicar a *View* com o *Model*, e realiza a lógica de negócio da aplicação. A *View* é a camada que efetivamente exibe as informações na tela e interage com o usuário. O *Model* é a camada que armazena as informações a serem exibidas. Ela pode interagir com a *ViewModel* tanto enviando dados, quanto recebendo, em um processo chamado de *two-way data binding*.

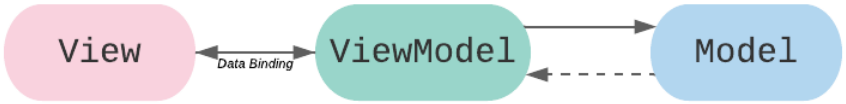

Figura 6 - Diagrama representando o padrão MVVM

O *framework* Vue.js[5] utiliza esse padrão MVVM e foi adotado na implementação do *front-end* da aplicação desenvolvida neste trabalho. Tendo sido projetado para ser adotado incrementalmente, com foco em componentização, *Vue.js* atende às necessidades deste trabalho e provê a estrutura inicial para o desenvolvimento da ferramenta. Ele funciona com base em um componente principal que inclui outros, formando assim uma árvore de componentes.

Além disso, esse *framework* apresenta a possibilidade de utilização do padrão SPAs (*Single Page Applications*). Com o SPA, a aplicação Web interage com o usuário reescrevendo a página atual dinamicamente, sem necessidade de recarregar novas páginas. Isso torna o uso da aplicação mais fluido, sem necessidade de requisições em todas as interações. Requisições específicas, que venham a ser necessárias, são feitas de maneira assíncrona, chamadas de AJAX (*Asynchronous JavaScript and XML*). Outro ponto positivo é que sua documentação é completa e sua comunidade é bem ativa, sendo possível encontrar auxílio para praticamente qualquer problema que possa aparecer e diminuindo a curva de aprendizado.

[5] **https://vuejs.org**

A implementação do *front-end* com o *framework Vue.js* neste trabalho utilizou as seguintes linguagens:

- ***Java Script*[6]:** Linguagem de programação base do *Vue.js*, tem como características ser interpretada e estruturada, fracamente tipada, multiparadigmas e assíncrona. Ela é usada para a comunicação entre as camadas do *framework*, assim como sua integração com ações no navegador.

- ***HyperText Markup Language* (HTML)[7]:** Linguagem de marcação interpretada utilizada na construção e organização dos componentes dentro de páginas web.

- ***Cascading Style Sheets* (CSS)[8]:** Linguagem de estilo utilizada para configurar a apresentação de componentes descritos em HTML, com foco em torná-los mais modernos e atrativos.

No Capítulo 3, será detalhado a implementação do padrão MVVM a partir da utilização do *Vue.js*.

2.5 Considerações Finais do Capítulo

Este capítulo apresentou fundamentos teóricos e tecnológicos relevantes para o desenvolvimento deste trabalho. No que diz respeito a aspectos teóricos, foram abordados tópicos relacionados a design de IHC, gerência de conhecimento e ontologias, que fornecem a base teórica para o desenvolvimento da ferramenta proposta neste trabalho. No âmbito dos aspectos tecnológicos, foram apresentadas as tecnologias envolvidas no processo de construção da ferramenta, descrevendo os *frameworks* que foram utilizados para implementar a arquitetura da ferramenta e linguagens usadas no seu desenvolvimento.

[6] **https://www.javascript.com**
[7] **https://html.spec.whatwg.org/multipage/**
[8] **https://www.w3.org/Style/CSS/**

Capítulo 3

KTID: Ferramenta de Apoio a Aspectos de Gerência de Conhecimento no Design de IHC

Este capítulo apresenta os principais resultados produzidos ao longo do desenvolvimento de KTID (Knowledge Tool for Interaction Design), uma ferramenta de apoio a aspectos de gerência de conhecimento no design de IHC. Na Seção 3.1 é apresentado o propósito de KTID, assim como os resultados produzidos durante a especificação e análise de requisitos da ferramenta. Na Seção 3.2 são apresentados resultados produzidos durante a fase de projeto da ferramenta. Na Seção 3.3 são apresentadas algumas telas da ferramenta. Por fim, a Seção 3.4 apresenta as considerações finais do capítulo.

3.1 Requisitos e Modelagem Conceitual

O levantamento de requisitos envolve um conjunto de atividades que deve permitir a comunicação, priorização, negociação e a colaboração com todos os interessados relevantes. Deve prover, ainda, uma base para o aparecimento, descoberta e invenção de requisitos, como parte de um processo altamente interativo (AURUM; WOHLIN, 2005). Nesta seção, são apresentados o propósito de KTID, os requisitos identificados, seus casos de usos, assim como o diagrama de classes da ferramenta.

3.1.1 Propósito da ferramenta

Como foi apresentado no Capítulo 1, a falta de uma conceituação comum sobre design de IHC tem sido um dos principais desafios envolvidos na aplicação de soluções de GC no design de IHC. HCIDO é uma ontologia de referência sobre design de IHC desenvolvida por Castro (2021), com o intuito de auxiliar no tratamento desses desafios. A partir da análise do modelo conceitual de HCIDO, observou-se que as representações físicas das escolhas de design não refletiam os mesmos relacionamentos que tinham suas correspondentes mentais (i.e., as representações físicas de *Mental HCI Design Choices*, *Mental HCI Design Components*, *Mental User Observable States* e *Mental User Input Resulting States* estão todas reduzidas a *HCI Design Choices*). Além disso, como *HCI Design Specifications* são agregações de *HCI Design Choices*, nem sempre é possível perceber facilmente todas as escolhas de design como partes da especificação de design

na qual elas estão codificadas, ou seja, *HCI Design Specifications* são vistas como um todo, e não como um conjunto de várias partes (CASTRO, 2021).

Diante disso, utilizando HCIDO como uma fonte de conhecimento sobre o domínio de design de IHC, o desenvolvimento de KTID tem o intuito de prover uma ferramenta que *auxilia designers de IHC a descrever, compartilhar e recuperar, de forma estruturada, informações relacionadas a escolhas de design feitas em especificações de design.* Desta forma, a utilização de KTID permite apoiar aspectos de gerência de conhecimento relativos ao design de IHC, tais como representação, armazenamento, acesso e avaliação de conhecimento.

3.1.2 Requisitos

Com base no conhecimento sobre o domínio de design de IHC provido por HCIDO e visando atender ao propósito da ferramenta descrito na seção anterior, foram levantados os requisitos, apresentados na Tabela 1.

Tabela 1 - Requisitos funcionais identificados para o KTID

ID	Requisito	Dependência
RF01	Deve ser possível manter[9] os objetos de design, com as informações referentes ao seu título e descrição.	
RF02	Deve ser possível manter requisitos de usuário que devem ser atendidos nos objetos de design, informando seu título, descrição.	RF01
RF03	Deve ser possível manter especificações de design, informando sua versão, tamanho de tela, tipo e arquivos de imagem.	RF01
RF04	Deve ser possível manter escolhas de design, relacionando-as ao designer responsável por ela, a um arquivo de imagem e a uma especificação de design, além do título e descrição da escolha.	RF03
RF05	Deve ser possível manter componentes de design, com seu título e descrição.	RF01
RF06	Deve ser possível associar escolhas de design a componentes de design aos quais elas se referem.	RF04, RF05

[9] A palavra "manter" é usada na descrição dos requisitos para referir-se a operações CRUD (i.e., inclusão, acesso, atualização e remoção)

RF07	Deve ser possível associar escolhas de design aos requisitos de usuário que as motivaram.	RF02, RF04
RF08	Deve ser possível manter avaliações de escolhas de design, com sua nota e descrição da avaliação.	RF04
RF09	Deve ser possível buscar escolhas de design registradas na ferramenta, apresentando e possibilitando a filtragem de informações relativas a seus detalhes (imagem da escolha, seu nome, seu componente de design, os requisitos do usuário associados, seu objeto de design e suas datas de criação e atualização), criador e avaliações.	RF04, RF08

3.1.3 Casos de Uso

Os casos de uso têm como propósito capturar e descrever as funcionalidades que um sistema deve prover para seus usuários (atores) (BARCELLOS, 2018). Com base na identificação dos *stakeholders* mais frequentemente envolvidos no design de IHC apresentada como resultado de uma investigação sobre GC no design de IHC (CASTRO, 2021), foram definidos os seguintes atores para a ferramenta KTID:

- **Designer de IHC:** Papel responsável pela criação das especificações de design que descrevem como aspectos de IHC do objeto de design devem ser implementados.
- **Gerente de Projetos:** O gerente de projetos tem como função acompanhar o andamento do desenvolvimento do objeto de design, assim como garantir que todos os requisitos de usuário sejam abordados nas especificações de design. Também é responsável por manter os usuários da ferramenta.
- **Desenvolvedor:** Tem como função implementar o que foi especificado pelo designer de IHC, precisando assim de acesso aos detalhes das escolhas de design para obter conhecimento de informações que não estão explícitas na especificação.

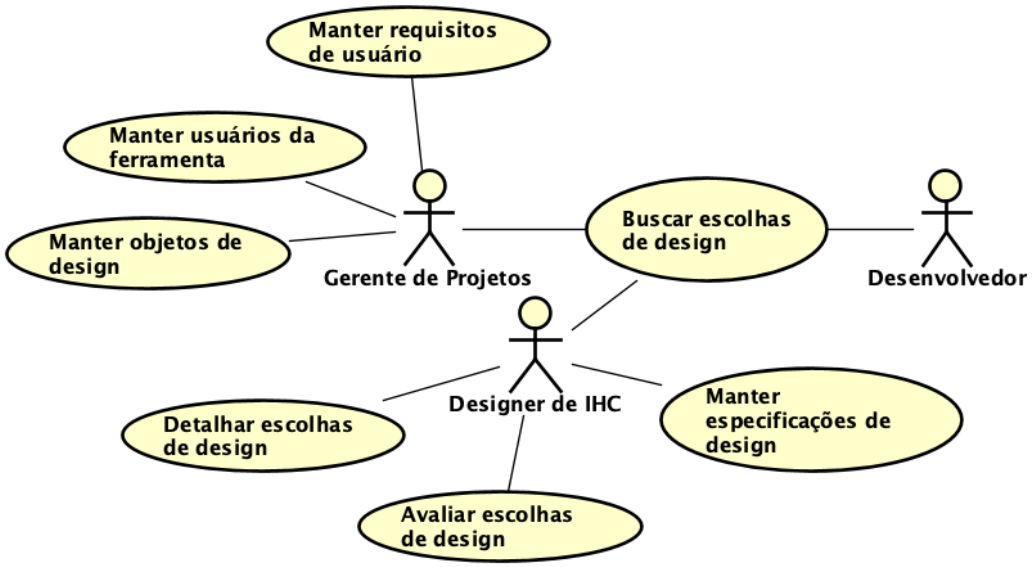

Figura 7 - Diagrama de casos de uso de KTID

Na Figura 7, foram apresentados os casos de usos do sistema. O caso de uso **Manter Usuários da Ferramenta** é realizado pelo Gerente de Projetos e tem como objetivo gerenciar o acesso dos demais atores à ferramenta. O acesso inicial do Gerente de Projetos deve ser provido através de um usuário padrão.

Os casos de uso cujos títulos começam com "Manter" (**Manter requisitos de usuários, Manter objetos de design e Manter especificações de design**) se referem a operações CRUD (*Create, Read, Update, Delete*). Devido ao curto período para o desenvolvimento e disponibilização da ferramenta para uso, alguns dos requisitos apresentados não englobam todas as operações CRUD, tendo sido priorizadas as operações primordiais. As operações pendentes ficaram como melhorias futuras da ferramenta. O objetivo desses casos de uso são incluir e manter os recursos que apoiam a realização das principais funcionalidades do sistema, que serão apresentadas a seguir.

O Designer de IHC é o papel com maiores responsabilidades. Ele é responsável por **Manter Especificações de Design** que especificam um objeto de design, incluindo informações sobre o número de versão da especificação, seu tipo (ex.: *wireframe*, *mockup*) e o tamanho de tela no qual o objeto de design será visualizado (ex.: celular, tablet, desktop). Além

disso, os artefatos que descrevem ou representam a especificação de design também podem ser adicionados. A partir dos artefatos da especificação de design, o Designer de IHC pode **Detalhar Escolhas de Design**, informando título, descrição dos detalhes que levaram à escolha e destacando qual a parte do artefato se refere aquela escolha. Além disso, pode-se informar o componente de design ao qual aquela escolha se relaciona, assim como os requisitos do usuário que a motivaram. Esse papel também pode **Avaliar Escolhas de Design**, permitindo assim que outras pessoas possam usar a avaliação como critério na consulta por escolhas anteriores que podem ser reaproveitadas em objetos de design futuros.

O Gerente de Projetos tem autonomia para **Manter os Objetos de Design** e para **Manter os Requisitos de Usuário**. Além disso, é sua responsabilidade **Manter Usuários da Ferramenta**, dando as permissões relevantes a cada cargo.

O caso de uso do Desenvolvedor é a **Busca das escolhas de design** de cada um dos projetos, que também pode ser realizada pelos dois outros atores, visando entender o processo de escolha do designer para ter uma maior compreensão e autonomia ao realizar a implementação da funcionalidade.

3.1.4 Diagramas de Classes

As classes de um modelo representam a expressão inicial do sistema. O paradigma de desenvolvimento orientado a objetos tem como atividade crucial na modelagem conceitual estrutural a identificação das classes do sistema (BARCELLOS, 2018). Como a ontologia de referência HCIDO (CASTRO, 2021) foi utilizada como base para entendimento do domínio de aplicação da ferramenta, algumas classes de KTID foram identificadas a partir da conceituação de HCIDO. A Figura 8 apresenta o diagrama de classes desenvolvido para a ferramenta KTID, destacando os conceitos de HCIDO relacionados a cada classe. Na figura, as classes em amarelo foram derivadas de conceitos de HCIDO e as classes em cinza foram modeladas a partir da especificação de requisitos de KTID. Notas verdes indicam conceitos de HCIDO que derivaram classes no modelo conceitual de KTID diretamente e notas roxas indicam conceitos de HCIDO e SEON que inspiraram a criação de novas classes. Por fim, as notas rosas descrevem restrições visando garantir a integridade dos dados.

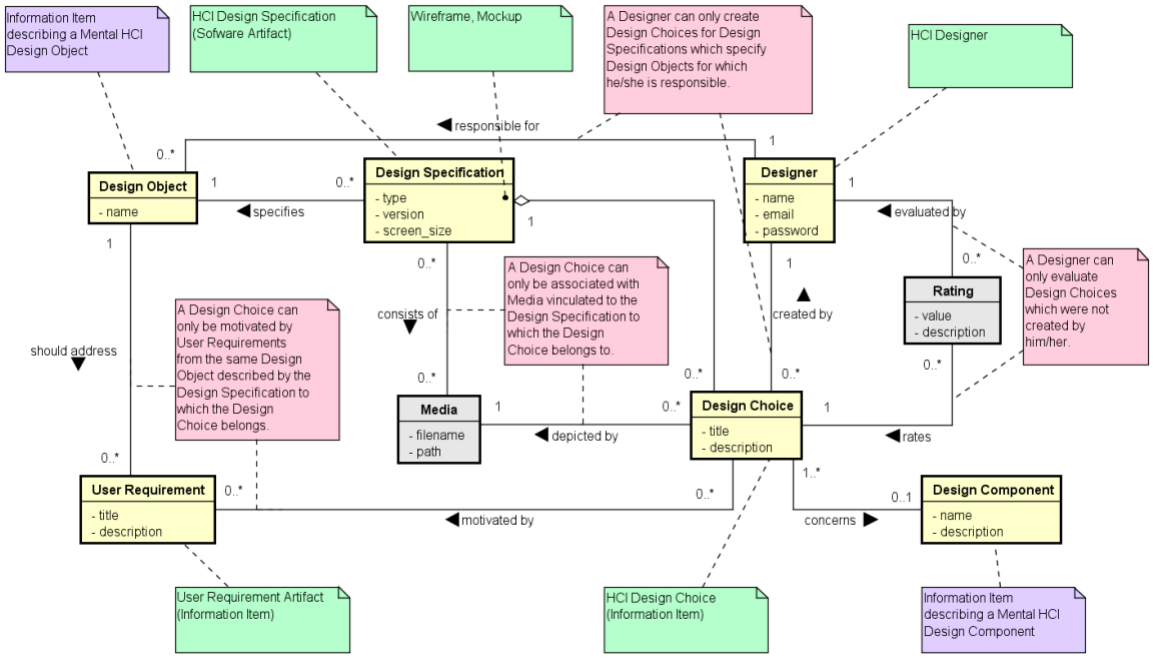

Figura 8 - Diagrama de classes de KTID e os conceitos correspondentes em HCIDO (CASTRO, 2021)

Para evitar conflitos semânticos do termo "*User*" com a conceituação de HCIDO (i.e., usuários do objeto de design), o termo dado à classe que representa os usuários da ferramenta foi **Designer**. Apesar do nome, ela se refere a quaisquer um dos atores apresentados anteriormente que podem utilizar o sistema. O termo **Designer** foi escolhido, portanto, por este ser o ator principal da ferramenta. O diagrama de classes será descrito com base na sua utilização por um designer de IHC, por seu amplo papel, que tem acesso a todos os recursos da ferramenta.

No diagrama, as classes ligadas às notas roxas foram necessárias para o registro de informações que apareciam na ontologia somente como aspectos mentais, mas na ferramenta esse conhecimento tem que ser explícito. Essas classes são **Design Component**, que se associa ao conceito de *Mental HCI Design Component*, e **Design Object** que se associa ao conceito de *Mental HCI Design Object*.

Em notas rosas, estão apresentadas restrições para garantir integridade dos dados. Na relação entre **Design Object**, **User Requirement** e **Design Choice**, temos o fato de as escolhas de design só poderem ser motivadas por requisitos de usuários provenientes do mesmo

objeto de design descrito pela especificação de design que a escolha de design pertence. Na relação entre **Design Specification** e **Design Choice** com a classe **Media**, temos como restrição o fato que um arquivo de imagem associado a uma escolha de design deve, necessariamente, estar de acordo com a mesma especificação de design. A restrição apresentada entre **Design Object**, **Designer** e **Design Choice** existe, pois um usuário apenas pode criar escolhas de design associadas aos objetos de design que foram designados para ele. Por fim, a restrição relacionada as classes **Designer**, **Design Choice** e **Rating** restringe que um usuário não pode avaliar suas próprias escolhas de design.

As classes **Media** e **Rating** estão destacadas na cor cinza, pois não correspondem diretamente a nenhum conceito da ontologia, contudo foram necessárias para o armazenamento de informações, como classes associativas para o registro de relações. A classe **Media** corresponde aos dados das imagens adicionadas a plataforma e a classe **Rating** é a responsável pelo armazenamento das avaliações das escolhas de design feitas pelos usuários da plataforma, segundo as restrições já apresentadas.

3.2 Projeto de Sistema

O objetivo da fase de projeto de software é produzir uma solução para o problema identificado e modelado durante o levantamento e análise de requisitos, incorporando a tecnologia aos requisitos e projetando o que será construído na implementação (BARCELLOS, 2018).

A ferramenta KTID foi desenvolvida utilizando-se as tecnologias apresentadas na seção 2.4. A seção a seguir apresenta a arquitetura de KTID.

3.2.1. Arquitetura de Software

Arquitetura de Software de um sistema computacional é a estrutura (ou estruturas) do sistema que compreende elementos de software, propriedades externamente visíveis desses elementos e os relacionamentos entre elas (BASS, CLEMENTS e KAZMAN, 2003).

KTID foi desenvolvida seguindo um paradigma cliente-servidor. No projeto da arquitetura da ferramenta, optou-se, portanto, por dividi-la em dois subsistemas, um que é executado no cliente e outro que é executado no servidor. O subsistema do servidor (*back-end*) foi desenvolvido utilizando o *framework* Laravel, que segue o padrão MVC, sendo composto pelas camadas de

lógica de negócio (CLN) e de gerência de dados (CGD), que correspondem, respectivamente aos componentes *Controller* e *Model* no padrão MVC. A CLN tem como responsabilidade as regras de negócio da ferramenta e foi desenvolvida seguindo o padrão Modelo de Domínio, sendo composta pelo Componente de Domínio de Problema (CDP), que será posteriormente detalhado. A CGD, por sua vez, trata a persistência dos dados no banco. O subsistema do cliente (*front-end*) foi implementado utilizando o *framework* Vue.js, que segue o padrão MVVM, sendo composto pela camada de interface com o usuário (CIU), que trata aspectos de exibição e interação com os usuários e corresponde aos componentes *View* e *ViewModel* no padrão MVVM. A CIU ainda se divide em um componente de interação humana (CIH) e um componente de controle de interação (CCI), que serão detalhados posteriormente. Sendo assim, a arquitetura de KTID combina os padrões MVC e MVVM de modo que o *Model* e *Controller* do MVC se complementem com o *ViewModel* e *View* do MVVM, permitindo assim possibilidades mais ricas de interação com a ferramenta. A Figura 9 apresenta um diagrama representando a divisão da arquitetura de KTID nos elementos descritos acima e nas próximas seções são apresentados mais detalhes de cada camada.

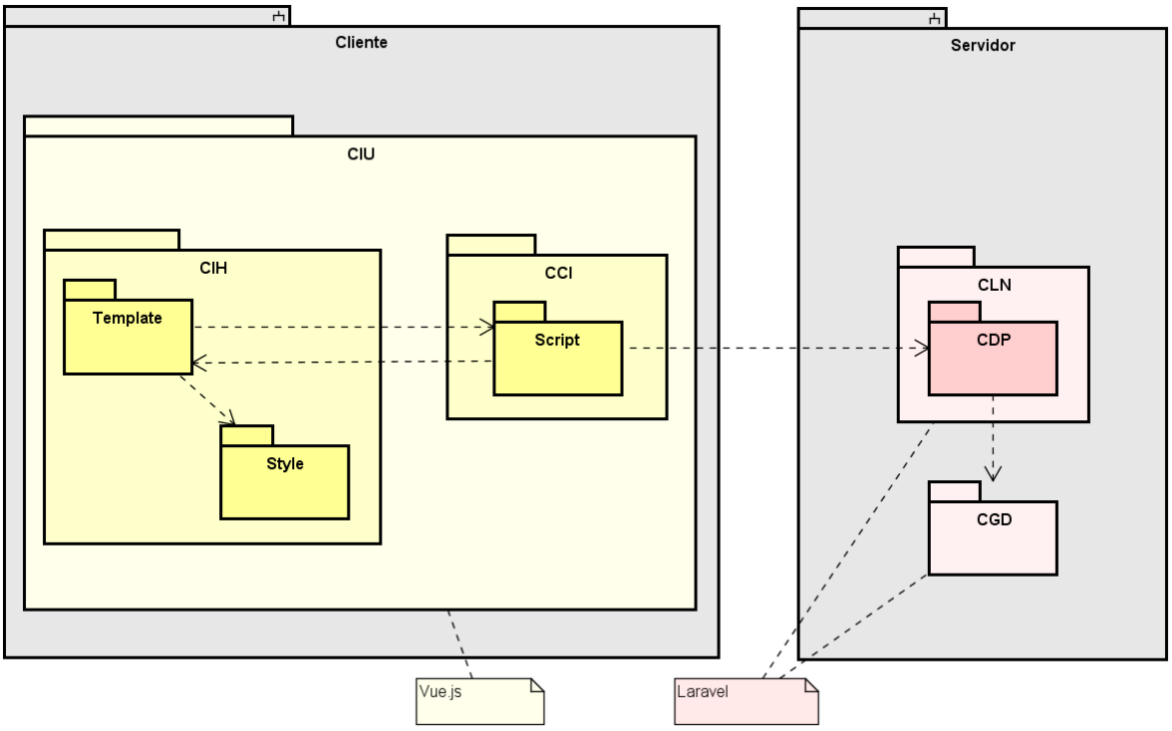

Figura 9 - Arquitetura de Software da ferramenta KTID

Nas próximas subseções são apresentados mais detalhes de cada camada, com fragmentos de diagramas do projeto.

3.2.1.1. Camada Lógica de Negócio

Com base no escopo e nas funcionalidades necessárias para a ferramenta, o padrão de desenvolvimento da camada de lógica de negócio escolhido foi o de Modelo de Domínio. Nesse padrão, as responsabilidades são distribuídas nos objetos de domínio do problema e a lógica de aplicação é pulverizada nesses objetos, sendo que cada objeto tem uma parte de lógica que é relevante a ele (BARCELLOS, 2018).

3.2.1.1.1 Componente de Domínio do Problema:

Essa camada é composta pelo Componente de Domínio do Problema, que são modelos conceituais estruturais produzidos na fase de análise do projeto. A Figura 10 apresenta o diagrama de classes produzido nesta fase.

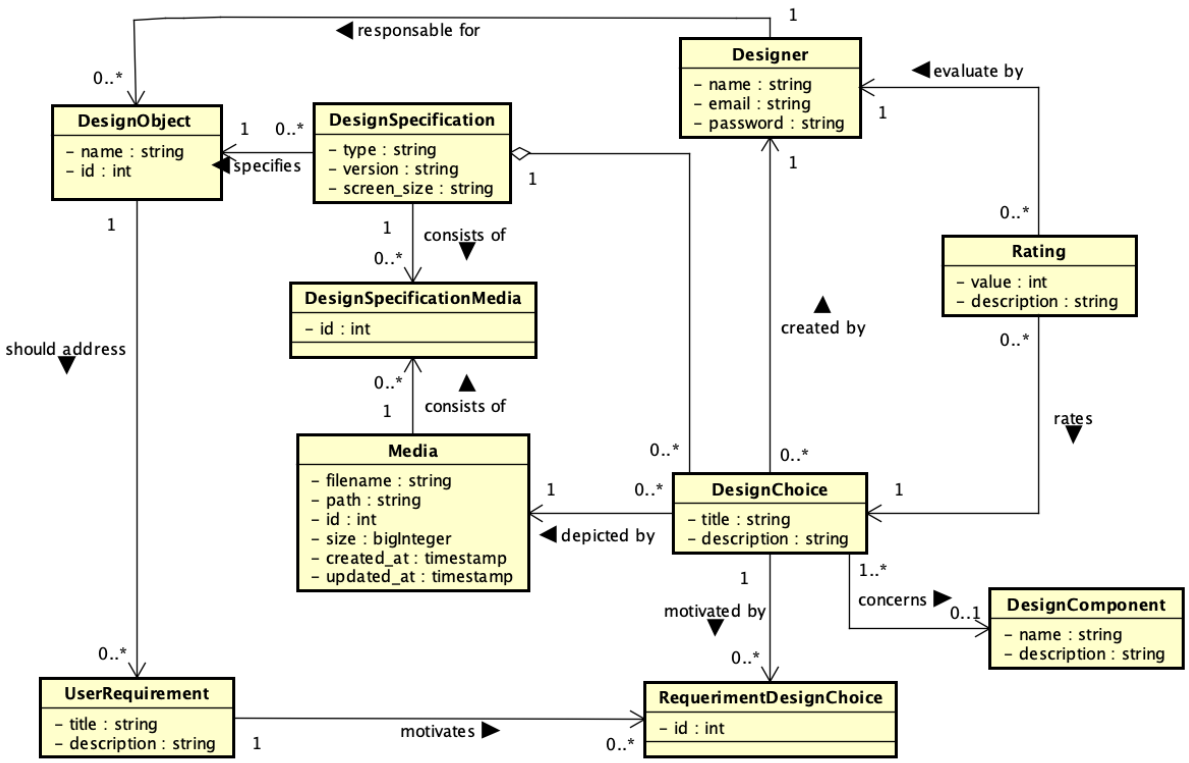

Figura 10 - Diagrama de Classes do CDP da ferramenta

Como o padrão adotado foi o de Modelo de Domínio, a CPD engloba toda a lógica de negócio, pois não há classes gerenciadoras de tarefas para controlar os casos de uso. Esse controle ocorre diretamente nos objetos de domínio, que recebe a parte lógica referente aos casos de uso de cada um deles.

3.2.1.2. Camada de Interface com o Usuário

Sistemas de informação são desenvolvidos para serem utilizados por pessoas. Assim, um aspecto fundamental no projeto é a interface com o usuário (IU). O projeto da IU estabelece uma forma de comunicação entre as pessoas e o sistema e como o sistema apresentará as informações a ele (BARCELLOS, 2018). Para isso, essa camada envolve dois componentes, o de Interação Humana e o de Controle de Interação.

Em KTID, a camada de interface com o usuário é implementada utilizando o *framework Vue.js*. Nele, cada arquivo implementa um componente de interface, que pode ser uma tela ou apenas partes de estruturas de telas (subcomponentes) a serem reutilizadas. Cada arquivo é

dividido em três partes: *Template* e *Style*, que implementam o Componente de Interação Humana (CIH) e o *Script*, que implementa o Componente de Controle de Interação (CCI). A seguir são detalhados mais detalhes de cada componente.

3.2.1.2.1 Componente de Interação Humana

O Componente de Interação Humana (CIH) corresponde à parte visual de um sistema, que compreende os objetos gráficos utilizados na interação com o usuário. É a implementação da interface do sistema propriamente dita e deve ser mantida separada do resto do sistema visando à manutenibilidade, já que o fluxo de alteração de interface num software costuma ser frequente.

O desenvolvimento desse componente iniciou-se com a prototipação da interface de algumas telas da ferramenta desenhadas a lápis em uma folha de papel, para facilitar o processo criativo.

Para facilitar a implementação da interface da ferramenta, foi utilizado o tema *Core.ui*[10], que provê um visual base e componentes como botões, campos de texto e menus já implementados forma a se adaptar a diferentes tamanhos de tela, tornando assim o acesso à ferramenta viável em dispositivos diferentes, como computadores, tablets e celulares.

O CIH compreende os subcomponentes *Template* e *Style*. O *Template* (Figura 11) consiste em código HTML e nele são declarados todos os elementos a serem exibidos na respectiva porção da tela, definindo suas posições e a chamada de métodos e dados fornecidos pelo CCI. Já o componente *Style* corresponde a códigos de estilização do *Template*, feitos em CSS. Buscando reúso e uma constância visual na ferramenta, toda a estilização foi feita em um arquivo separado e centralizado. Além disso, são herdadas definições de estilo do *Core.ui* para os elementos básicos, como botões, formulários e tabelas.

[10] **https://coreui.io/**

```
<template>
  <CRow>
    <CCol col="12" xl="12">
      <CCard>
        <CCardHeader>
          <h6>{{ projectName }}</h6>
          <h4>
            Requirement - {{ userRequirementTitle }}
          </h4>
          <p style="white-space: pre;">{{ userRequirementDescription }}</p>
        </CCardHeader>
        <CCardBody>
          <CRow>
            <CCol col="12">
              <DesignChoicesGrid :requirementId="userRequirementId" />
            </CCol>
            <CCol col="12" align="right">
              <CButton  color="primary" @click="selectDesignChoice()">Add new design choice</CButton>
            </CCol>
          </CRow>
        </CCardBody>
      </CCard>
    </CCol>
  </CRow>
</template>
```

Figura 11 - Trecho de código do componente *template* do *vue.js*

3.2.1.2.2 Componente de Controle de Interação

O CCI é o componente responsável pela comunicação entre as regras de lógica de negócio e a interface do usuário. No Vue.js, ele é composto pelo subcomponente *Script* (Figura 12), que implementa todos os métodos referentes a um componente ou tela. Nele está contida a estrutura que abstrai os dados da CGD, os métodos que conectam as ações dos usuários no CIH às ações correspondentes do CLN através de uma API (*Application Programming Interface*), atualizando o que é exibido no *Template* quando necessário.

```
<script>
import axios from 'axios'
import DesignChoicesGrid from '../base/DesignChoicesGrid'

export default {
  name: 'DesignChoices',

  components: {
    DesignChoicesGrid,
  },

  data: () => {
    return {
      projectId: 0,
      projectName: '',
      userRequirementTitle: '',
      userRequirementDescription: '',
      userRequirementId: 0,
    }
  },
  methods: {
    selectDesignChoice( ) {
      const newDesignChoice = `/projects/` + this.$route.params.projectId + `/requirements/`
          + this.$route.params.requirementId + '/newDesignChoice'
      this.$router.push({path: newDesignChoice})
    },
    getRequerimentInfo() {
      let self = this;
      axios.get('/api/userRequirement/showRequirement?id=' + self.userRequirementId + '&token='
            + localStorage.getItem("api_token"))
      .then(function (response) {
        self.userRequirementTitle = response.data.title;
        self.userRequirementDescription = response.data.description;
      }).catch(function (error) {
        console.log(error);
      });
    },
    getProjectInfo() {
      let self = this;
      axios.get('/api/projects/show?id=' + self.projectId + '&token='
            + localStorage.getItem("api_token"))
      .then(function (response) {
        self.projectName = response.data[0].name;
      }).catch(function (error) {
          console.log(error);
      });
    },
  },

  mounted: function(){
    let self = this;
    self.projectId = self.$route.params.projectId;
    self.userRequirementId = self.$route.params.requirementId;
    self.getProjectInfo();
    self.getRequerimentInfo();
  }
}
</script>
```

Figura 12 - Trecho de código do componente *script* do *vue.js*

3.2.1.3. Camada de Gerência de Dados

O banco de dados da aplicação é relacional (MySQL) e é acessado através do *framework Laravel*, que provê a infraestrutura de persistência para a aplicação. O mapeamento objeto relacional do *Laravel* é feito através do *Eloquent*, que utiliza o padrão de Registro Ativo (*Active Record*), visando à simplicidade. O mapeamento dos elementos do banco de dados para o modelo é feito através de uma convenção na nomenclatura de arquivos, classes e tabelas, tornando mais fluido o processo de desenvolvimento. Além disso, os métodos da CLN têm visibilidade e acesso aos dados do modelo de forma enxuta e legível através da utilização do padrão de Fachadas, o que faz com que o acoplamento seja reduzido.

3.3 A Ferramenta KTID

Nesta seção, são apresentadas as telas da ferramenta, assim como uma descrição do fluxo de utilização dela. Na descrição, o termo "usuário" será utilizado genericamente para referir-se à pessoa que está utilizando a ferramenta naquele momento, independente do papel que ela assume de acordo com os atores dos casos de uso.

Ao acessar o endereço web[11] no qual a ferramenta está hospedada, o ponto de partida é fazer a autenticação. Através dela, é possível identificar o autor das operações realizadas na ferramenta.

[11] http://bit.ly/KTID-tool

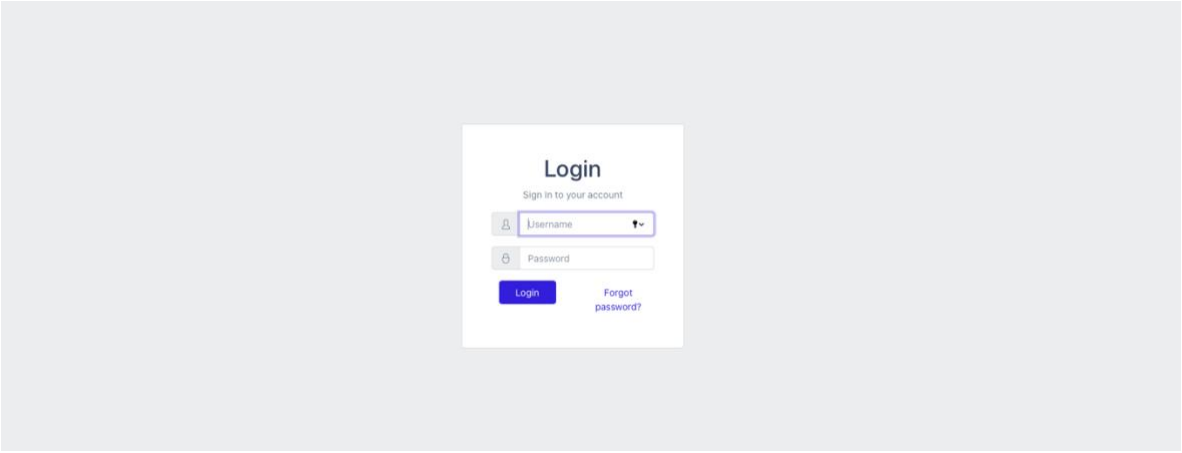

Figura 13 - Tela de Login de KTID

Após o login (Figura 13), a primeira tela exibida (Figura 14) é uma lista contendo todos os objetos de design cadastrados, com os botões para acessar os respectivos requisitos e especificações de design. Nessa tela há também um botão referente a inclusão de novos objetos de design (*Add a new design object*). Em KTID um objeto de design pode se referir, por exemplo, ao design de um sistema como um todo ou de parte de um sistema.

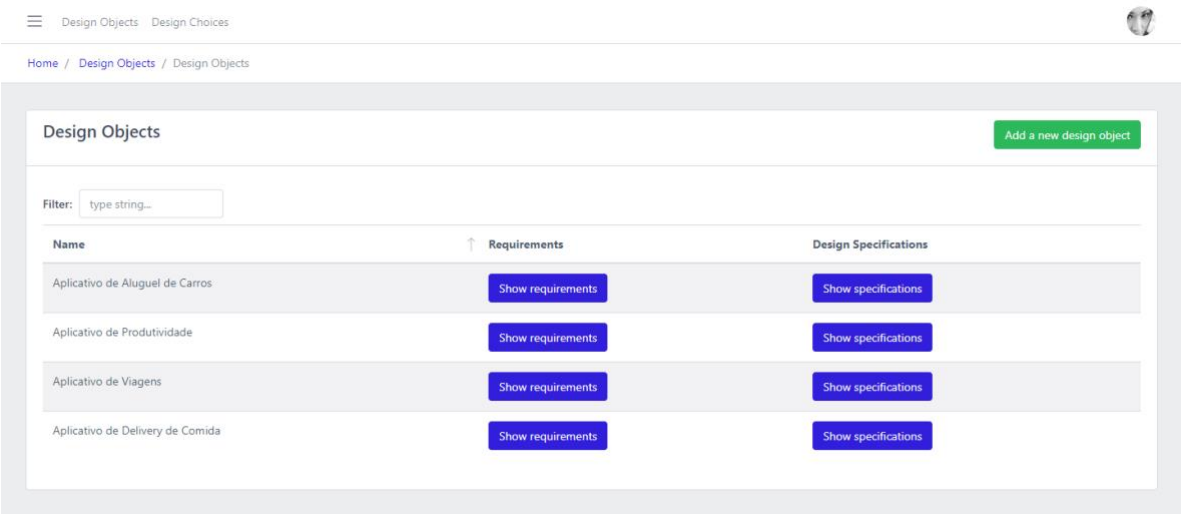

Figura 14 - Lista de objetos de design cadastrados

Ao cadastrar um novo objeto de design (Figura 15), informa-se seu nome e o usuário pode escolher entre apenas salvá-lo e voltar para lista de objetos de design (*Just Save*) ou salvá-lo e em seguida adicionar os requisitos de usuário que esse objeto de design deve atender (*Save and Add Requeriments*).

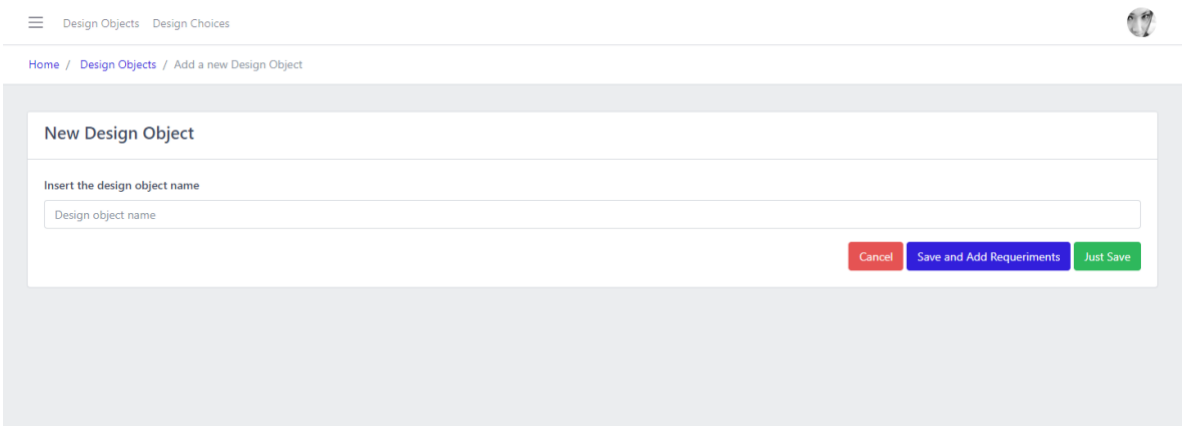

Figura 15 - Cadastro de novos objetos de design

Caso o usuário escolha adicionar requisitos de usuário na sequência do cadastro de objetos de design, é apresentada a tela de cadastro de requisitos de usuário (Figura 16). Nessa tela, são exibidos campos para inserção do título do requisito e sua descrição, além de uma tabela ao lado com todos os requisitos de usuário já adicionados para o objeto de design em questão. Ao inserir as informações, pode-se finalizar a tarefa apenas salvando o requisito de usuário e voltar para a lista de requisitos (*Just Save*), ou, caso haja mais requisitos de usuário a serem adicionados, o usuário pode salvar o último requisito inserido e inserir um novo (*Save and add more*). Com isso, os campos de valores são limpos e a tabela exibida na direita é atualizada, exibindo também os últimos requisitos de usuário inseridos.

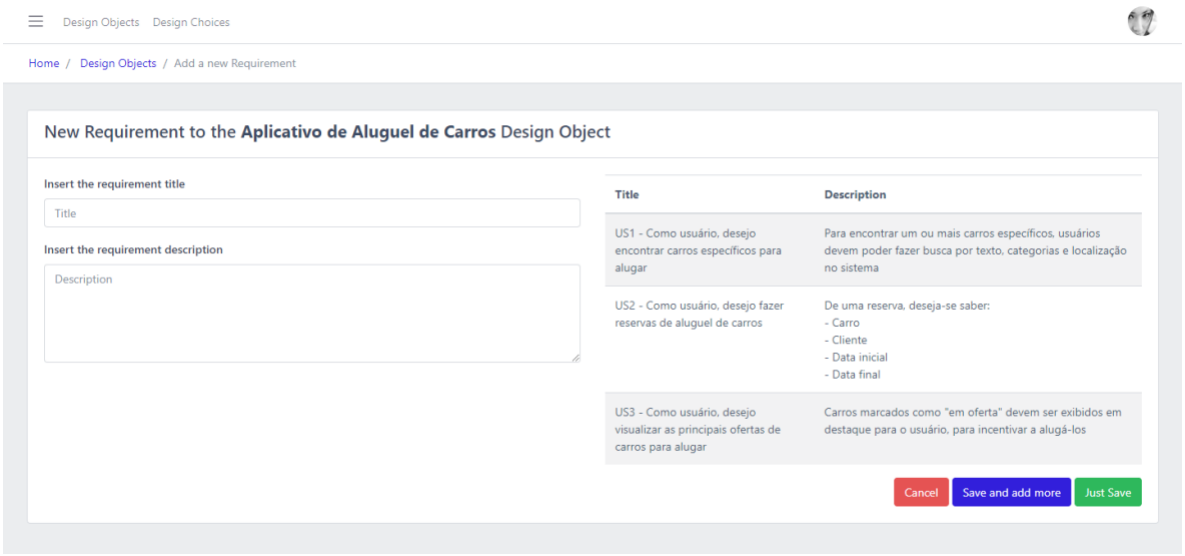

Figura 16 - Cadastro de novos requisitos do usuário para um objeto de design

Ao clicar no botão que apenas salva (*Just Save*) da Figura 16, ou ao clicar em mostrar requisitos de usuário (*Show Requirements*) de um objeto de design (Figura 14), o usuário é direcionado para a tela que lista todos os requisitos de usuário que devem ser considerados no objeto de design em questão (Figura 17). Nessa tela, há um link para adicionar novos requisitos de usuário (botão *Add a new requeriment*), que direciona para a tela da Figura 16. Além disso, também é possível editar os requisitos de usuário existentes (botão *Edit*) e ver escolhas de design associadas ou associar novas escolhas de design a cada requisito listado na tabela (botões *Show* e *Create*, respectivamente).

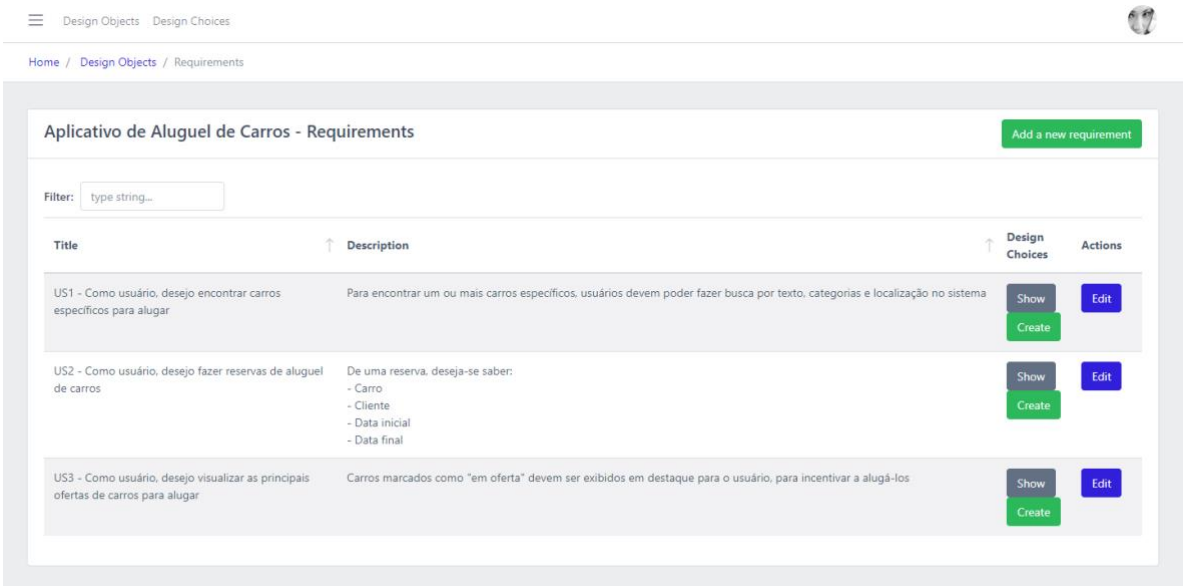

Figura 17 - Lista de requisitos de usuário cadastrados para um objeto de design

Ao clicar no botão para mostrar especificações (*Show specifications*) da Figura 14, o usuário é direcionado para a tela que exibe os detalhes das especificações de design do objeto de design em questão (Figura 18). Na barra superior, há um botão de listagem (*dropdrown*) com todas as versões de especificações de design já cadastradas para aquele objeto. Por padrão, exibe-se incialmente a versão mais recente. Quando o usuário altera a versão no *dropdown*, a página é recarregada com as informações correspondentes à versão da especificação selecionada.

O conteúdo de uma especificação de design é exibido em duas seções. Na seção inferior são exibidas as escolhas de design já identificadas nessa especificação. Ao clicar em uma escolha de design, o usuário é direcionado a tela contendo os detalhes dessa escolha, que será apresentada mais à frente. Já a seção superior apresenta os arquivos de imagens (*Media*) que descrevem a

especificação de design, assim como um botão para adicionar mais arquivos. Cada uma das imagens pode ser removida (botão *Delete*), visualizada (botão *View*) e selecionada para a identificação de uma nova escolha de design a partir dela (botão *Describe a Design Choice*).

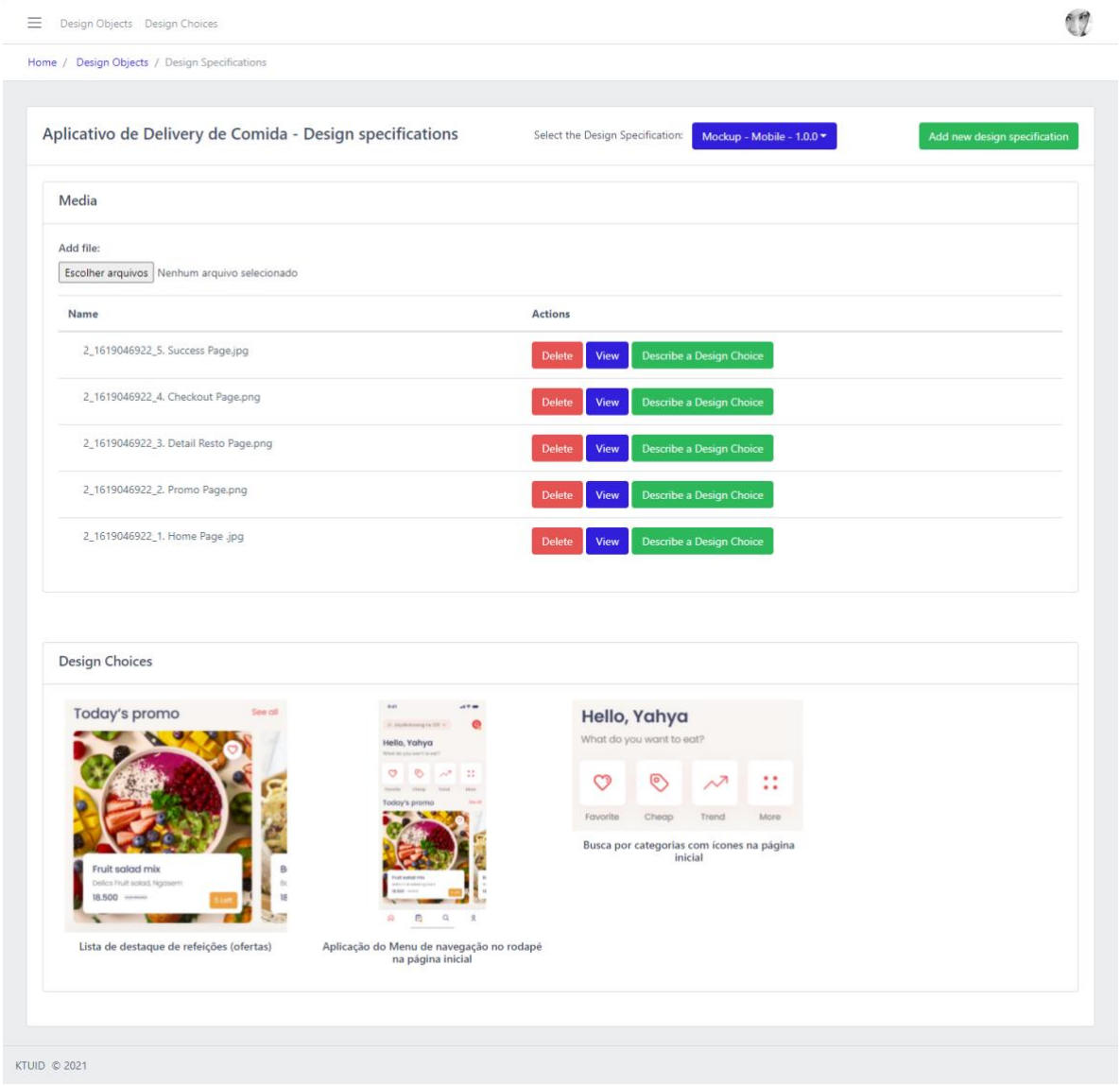

Figura 18 - Detalhes de uma especificação de design

À direita do *dropdown* de escolha da versão da especificação a ser exibida, há um botão para adicionar novas especificações de design (*Add new design specification*). Ao clicar nesse botão, o usuário é direcionado para a tela de inclusão de nova especificação de design (Figura 19). Nessa tela, primeiramente deve-se informar o tamanho de tela (*screen size*) que aquela versão da especificação de design busca contemplar, tendo como opções celular, computador, tablet e

desktop. Além disso, informa-se o tipo daquela versão (*version type*), indicando o nível de fidelidade e refinamento que o objeto de design é descrito na especificação (ex.: um *mockup* possui nível maior de fidelidade do que um *wireframe*). Por fim, deve-se informar o número da versão da especificação, com o objetivo de manter de forma simples um controle de versionamento e histórico de alterações nas especificações de design.

Figura 19 - Cadastro de nova especificação de design

Na Figura 19, ao clicar no botão de descrever uma nova escolha de design a partir de um arquivo de imagem da especificação (botão *Describe a Design Choice*), o usuário é direcionado para a tela de detalhamento de escolhas de design (Figura 20). Nessa tela, deve-se informar o título e descrição da escolha, assim como pode-se associar um componente de design a ela, podendo este ser um componente previamente existente ou um novo criado naquele momento. Além disso, ao clicar no botão *Crop Image*, a imagem é exibida num *pop-up*, possibilitando que o usuário selecione e corte na imagem uma região específica à qual a escolha de design se refere. Também é possível detalhar escolhas de design a partir da lista de requisitos de usuário (Figura 17), clicando no botão criar (*Creat*) da coluna referente a escolhas de design (*Design Choices*). Nesse caso, a escolha de design já é previamente associada ao requisito e ele é exibido junto as informações de especificação de design e objeto de design, a direita da tela. No momento de salvar a escolha de design, o usuário pode salvar e em seguida associar novos requisitos de usuário àquela escolha (*Save and associate requirements*), sendo assim direcionado para a tela que contém todos os requisitos de usuário associados ao objeto de design em questão, que podem ser marcados ou desmarcados um a um (Figura 21). Alternativamente, o usuário pode apenas

salvar (botão *Just Save*) a escolha, sendo então direcionado de volta para a tela de especificações desse objeto de design (Figura 18).

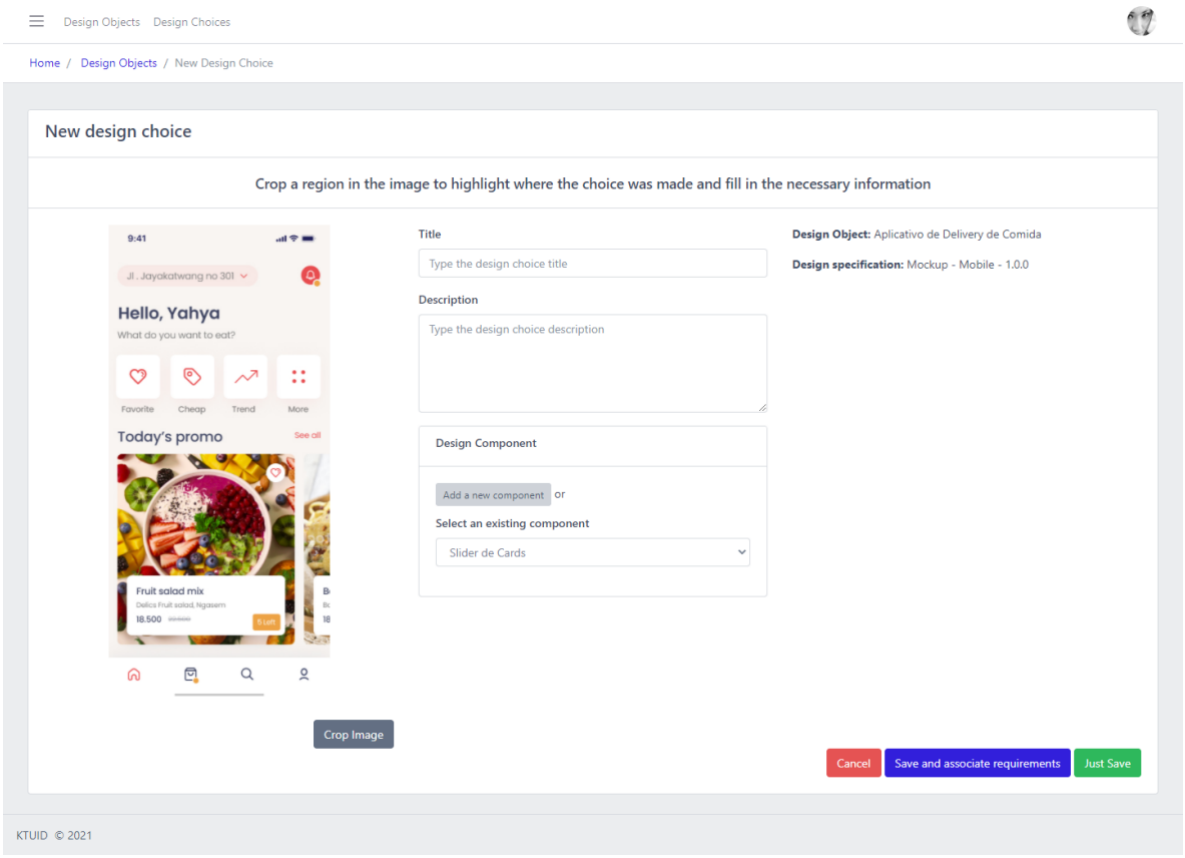

Figura 20 - Detalhando uma escolha de design

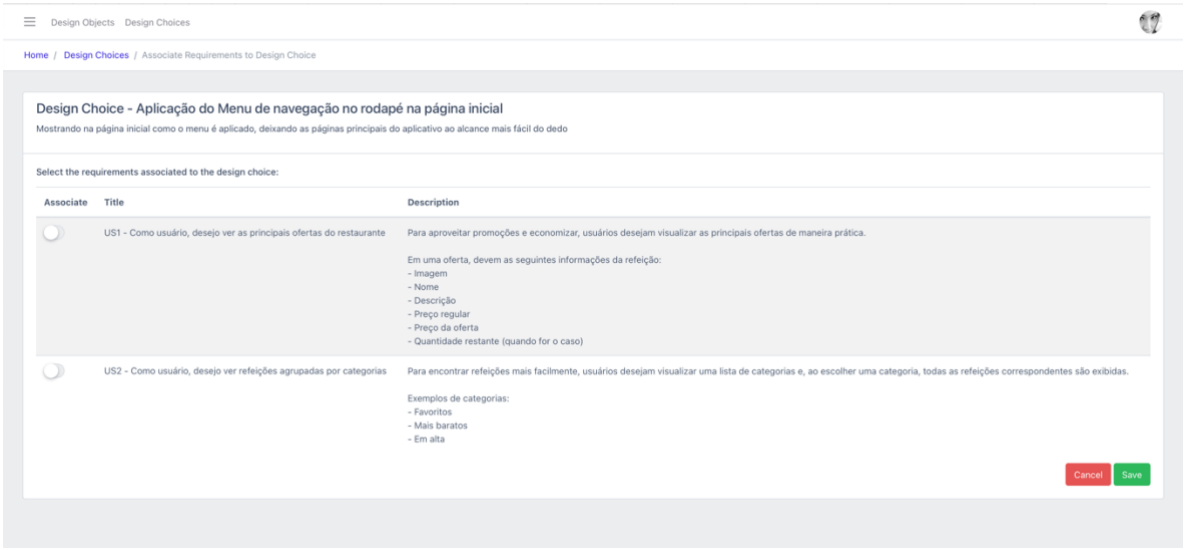

Figura 21 – Associando novos requisitos de usuário a uma escolha de design

Ao clicar em uma escolha de design dentre as listadas na tela de especificações de design (Figura 18), o usuário é direcionado para uma tela contendo todos os detalhes daquela escolha, como sua descrição, requisitos de usuário e componente de design associado (Figura 22). Ainda na Figura 22, no lado direito da tela é apresentada a nota média dada nas avaliações dessa escolha de design e um botão para visualizar os detalhes dessas avaliações (Figura 23). Caso a escolha ainda não tenha avaliações, é exibido um *pop-up* solicitando que o usuário avalie, com um botão (*Rating!*), que direciona para a tela de avaliação (Figura 25). Além disso, há um botão (*Edit*) no canto superior esquerdo, que, quando clicado, direciona o usuário para a tela de edição da escolha de design.

Na tabela de requisitos de usuário associados à escolha de design, é possível acessar, para cada requisito, a lista com todas as escolhas de design associadas àquele requisito (botão *Show all Design Choices*). Além disso, é possível associar novos requisitos de usuário (botão *Associate requirements*) a essa escolha de design, redirecionando o usuário para a tela apresentada na Figura 16.

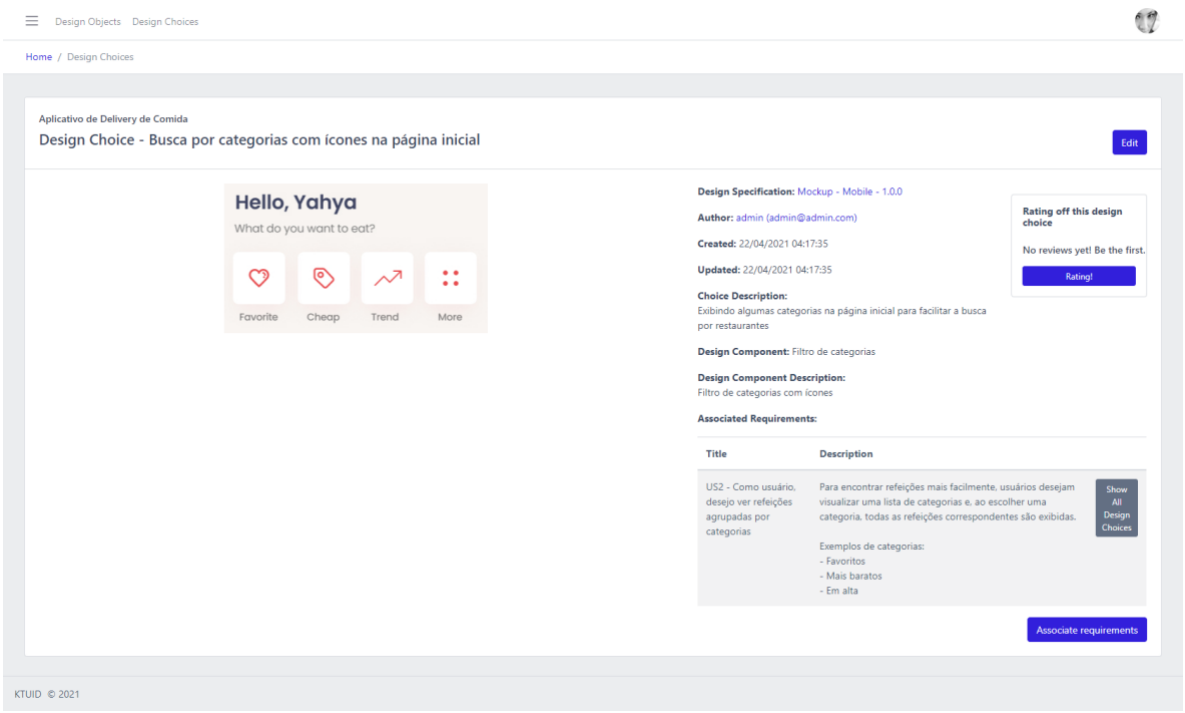

Figura 22 - Detalhes de uma escolha de design

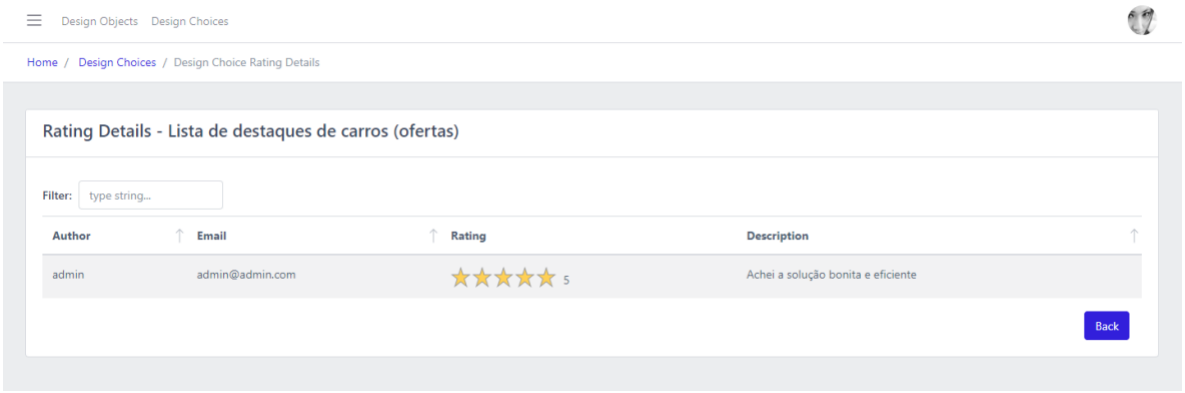

Figura 23 - Avaliações feitas para uma escolha de design

A Figura 24 apresenta a tela de busca de escolhas de design. Nela, são exibidos os detalhes contextuais de todas as escolhas já identificadas na ferramenta, como por exemplo requisitos de usuário e componentes relacionados, objeto de design e especificações aos quais elas se referem, além de outros detalhes, como a nota das avaliações previamente feitas. Na tabela, há um botão (*Show Details*) que direciona para a tela de detalhes da escolha, já apresentada anteriormente (Figura 22). Nesse momento, também é possível avaliar a escolha de design, ao

clicar no último campo da tabela, o botão *Rating*. Esse botão leva para a mesma tela de avaliação (Figura 25) que também é possível de ser acessada pela tela de detalhes da escolha de design (Figura 22).

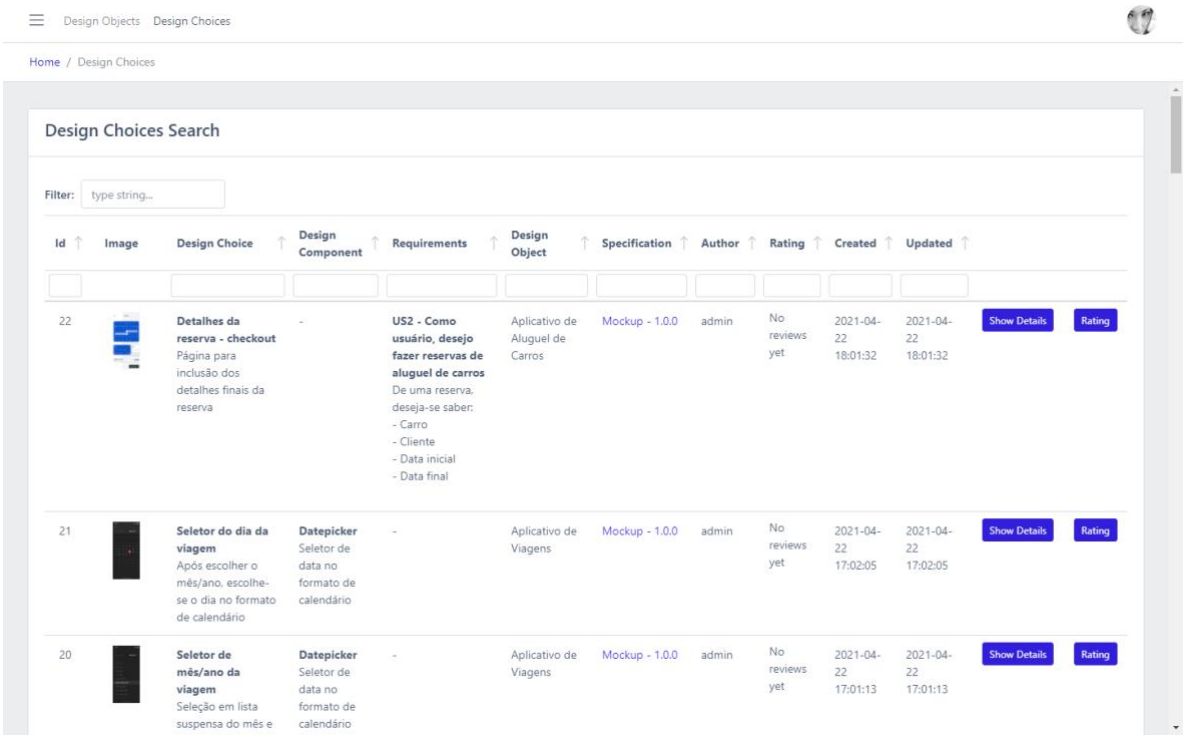

Figura 24 - Busca de escolhas de design

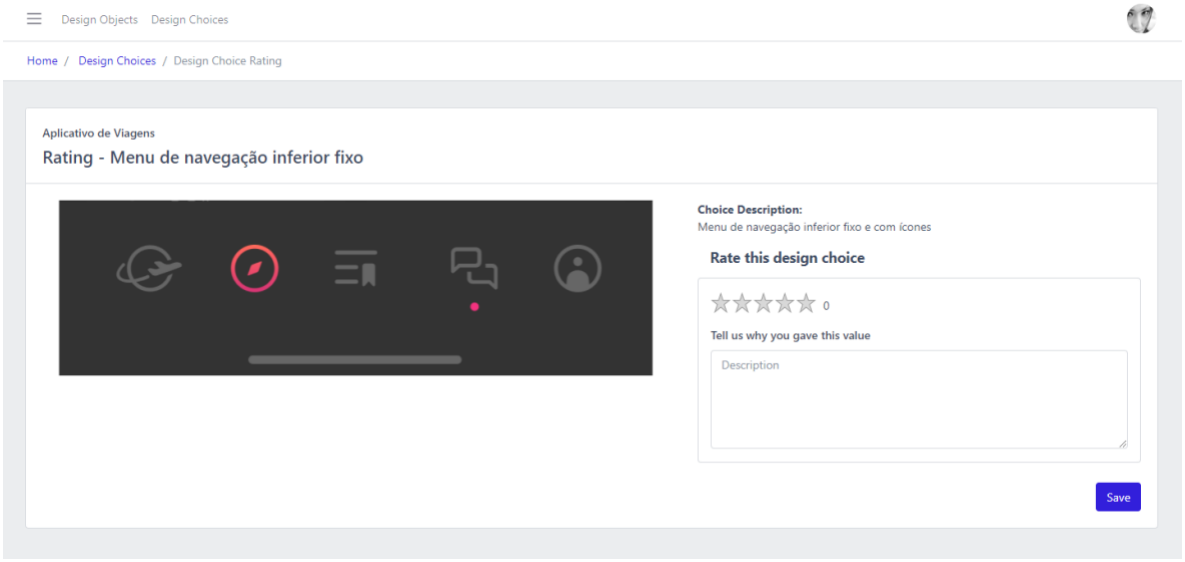

Figura 25 - Avaliando uma escolha de design

3.4 Considerações Finais do Capítulo

Este capítulo apresentou KTID, uma ferramenta de apoio a aspectos de gerência de conhecimento no design de interação humano-computador, apresentando seus requisitos e sua modelagem conceitual, seguindo com detalhes do projeto da ferramenta e finalizando com a apresentação das telas da ferramenta implementada. No próximo capítulo, são apresentadas as considerações finais do trabalho.

Capítulo 4
Considerações Finais

Neste capítulo são realizadas as considerações finais deste trabalho, sendo apresentadas suas principais contribuições e perspectivas de trabalhos futuros.

4.1 Conclusões

Por ser um processo intensivo em conhecimento, o design de IHC pode se beneficiar de princípios e práticas da Gerência de Conhecimento, adotando mecanismos efetivo que auxiliem na criação colaborativa e no compartilhamento de conhecimento sobre aspectos de IHC, tais como usuários, sistemas interativos, propósitos de uso, contextos de uso e design (CASTRO et al., 2020).

Neste trabalho, foi desenvolvida a ferramenta KTID, que tem como propósito apoiar aspectos de gerência de conhecimento no design de IHC. KTID foi desenvolvida a partir da conceituação de HCIDO (CASTRO, 2021), uma ontologia de referência sobre design de IHC, e permite a representação, armazenamento, recuperação e avaliação de conhecimento relativo a escolhas feitas por designers em especificações de design de IHC.

Ao longo do desenvolvimento do projeto, foram realizadas inúmeras práticas aprendidas durante a graduação em diferentes disciplinas. Os conhecimentos mais utilizados foram obtidos através das seguintes disciplinas:

- Engenharia de Software, onde foi apresentado todo o conhecimento necessário para o levantamento de requisitos e modelagem de casos de uso, o que tornou o trabalho a ser feito mais claro e preciso, facilitando a tomada de decisões de implementação. Também facilitou a organização do tempo para desenvolvimento, por prover previsibilidade quanto às funcionalidades a serem desenvolvidas.
- Programação II e III, que deram a base de conhecimentos essenciais para a implementação da ferramenta, tanto com relação à lógica de programação em si, quanto em relação aos paradigmas de programação Orientada a Objeto e Estruturada, fortemente utilizados durante a implementação. Além disso, o conhecimento prévio adquirido com as linguagens utilizadas nessas disciplinas atenuou a curva de

aprendizagem das linguagens de programação utilizadas na implementação da ferramenta (PHP e *JavaScript*), que foram aprendidas para o desenvolvimento deste trabalho.

- Bancos de Dados, que proveu conhecimento de base com relação a estruturação, modelagem e utilização de persistência em bancos relacionais, como o utilizado nesse sistema.

No Capítulo 1 da monografia foram apresentados os objetivos a serem alcançados no desenvolvimento deste trabalho. Na Tabela 2, são apresentados novamente cada um dos objetivos, assim como a indicação de atendimento e os resultados alcançados que evidenciam o atendimento dos mesmos.

Tabela 2 - Objetivos e sua situação na conclusão da monografia

Objetivo	Status	Resultado
Identificar itens de conhecimento relevantes ao design de IHC para serem tratados na ferramenta.	Atendido	Seleção de conceitos de HCIDO que foram aproveitados no desenvolvimento de KTID (vide Seção 2.3.1).
Elaborar a modelagem conceitual da ferramenta a partir de uma ontologia de referência sobre design de IHC.	Atendido	Especificação de requisitos e modelos de casos de uso e de classes de KTID (vide Seção 3.1)
Definir e projetar uma arquitetura para a ferramenta, levando em considerações padrões, modelos e paradigmas utilizados no projeto de sistemas de software;	Atendido	Elaboração do projeto da arquitetura e projeto detalhado de KTID (vide Seção 3.2).
Implementar, testar e disponibilizar para uso uma versão inicial da ferramenta.	Atendido	Ferramenta KTID implementada (vide Seção 3.3).

Entre as dificuldades para o desenvolvimento desse trabalho, destaca-se a compreensão de ontologias, assunto com o qual a autora nunca havia tido contato formal antes de iniciar os estudos para o desenvolvimento deste projeto. Além disso, o aprendizado dos *frameworks* utilizados foi extremamente enriquecedor, porém demandou um tempo considerável.

Vale ressaltar também que a versão inicial de KTID apresentada neste trabalho possui algumas limitações, como a ausência de algumas operações CRUD (edição e remoção) em alguns elementos da ferramenta. Tais limitações ocorreram devido a restrições de tempo para conclusão do trabalho e consequente priorização de funcionalidades para a disponibilização da ferramenta para uso. Desta forma, as limitações de KTID podem ser abordadas em trabalhos futuros, melhorando a ferramenta para o seu uso na prática.

Também vale destacar que KTID foi disponibilizada para uso de designers em um estudo experimental para avaliação da ferramenta. Os resultados do estudo forneceram indícios preliminares de que o uso de KTID é viável e que ela é útil no design de IHC, particularmente para designers menos experientes. No entanto, para designers mais experientes, resultados do estudo indicam que KTID pode influenciar na diminuição do processo criativo devido ao reúso de soluções já criadas. O estudo também apontou algumas necessidades de melhorias, que poderão ser tratadas em trabalhos futuros. Informações sobre o estudo de avaliação de KTID encontram-se em (CASTRO, 2021).

4.2 Trabalhos Futuros

Como trabalhos futuros, os seguintes pontos foram identificados como melhorias na ferramenta:

- Evoluir o cadastro de usuários da plataforma, implementando controle de acesso baseado em papéis.
- Implementar melhorias de usabilidade na ferramenta, deixando a interface e a interação mais fluidas e com melhores formas de exibição em caso de erros aos usuários.
- Evoluir as funcionalidades de edição, tornando o sistema mais flexível com a opção de editar mais informações. Implementar também outras operações de *CRUD* (ex.: remover objetos de design).

- Criar funcionalidades de gerenciamento para o Gerente de Projetos (ex.: acompanhar o andamento das definições de um designer em um determinado projeto).
- Criar funcionalidades de atribuição de escolhas de designs aos desenvolvedores responsáveis pela implementação do objeto de design.

Referências Bibliográficas

AURUM, Aybüke, WOHLIN, Claes, Engineering and Managing Software Requirements, Springer-Verlag, 2005

BARCELLOS, Monalessa P.: Engenharia de Software: Notas de aula. 2018. Departamento de Informática, Universidade Federal do Espírito Santo, Vitória-ES. Disponível em: < https://nemo.inf.ufes.br/wp-content/uploads/Monalessa/EngSoftware/NotasDeAula-EngSw-EngComp-v2018.pdf >. Acesso em: 8 maio 2018.

BASS, L., CLEMENTS, P., KAZMAN, R., Software Architecture in Practice, Second edition, Addison Wesley, 2003

CARROLL, John Millar. Human Computer Interaction (HCI). *In*: SOEGAARD, Mads; DAM, Rikke Friis (org.). The Encyclopedia of Human-Computer Interaction. 2nd. ed. Aarhus, Denmark: The Interaction Design Foundation, 2014. p. 21–61.

CHAMMAS, Adriana; QUARESMA, Manuela; MONT'ALVÃO, Cláudia. A Closer Look on the User Centred Design. **Procedia Manufacturing**, *[S. l.]*, v. 3, p. 5397–5404, 2015. DOI: https://doi.org/10.1016/j.promfg.2015.07.656.

CASTRO, Murillo Vasconcelos H. B: An Ontology to support Knowledge Management Solutions for Human-Computer Interaction Design, Dissertação de Mestrado, Departamento de Informática, Universidade Federal do Espírito Santo, 2021.

CASTRO, Murillo Vasconcelos H. B.; COSTA, Simone Dornelas; BARCELLOS, Monalessa P.; FALBO, Ricardo de A. Knowledge management in human-computer interaction design: A mapping study. *In*: 23RD IBEROAMERICAN CONFERENCE ON SOFTWARE ENGINEERING, CIBSE 2020 2020, **Anais** [...]. [s.l: s.n.]

COSTA, Simone Dornelas; BARCELLOS, Monalessa P.; FALBO, Ricardo de Almeida; CASTRO, Murillo Vasconcelos Henriques Bittencourt. Towards an Ontology Network on Human-Computer Interaction. *In*: (Gillian Dobbie, Ulrich Frank, Gerti Kappel, Stephen W. Liddle, Heinrich C. Mayr, Org.)CONCEPTUAL MODELING 2020, Cham. Anais [...]. Cham: Springer International Publishing, 2020. p. 331–341. Disponível em:

<http://link.springer.com/10.1007/978-3-030-62522-1_24>.

COSTA, Simone Dornelas: An Ontology Network to support Knowledge Representation and Semantic Interoperability in the HCI Domain. 2021. Federal University of Espírito Santo, Vitória – ES, Brazil., [S. l.], 2021.

DE SOUZA, Clarisse Sieckenius: The Semiotic Engineering of Human-Computer Interaction. The MIT Press, Cambridge, Massachussetts (2005)

DIX, Alan; DIX, Alan John; FINLAY, Janet; ABOWD, Gregory D.; BEALE, Russell. Human-computer interaction. [s.l.] : Pearson Education, 2003.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John M.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1994. ISBN 0201633612.

GUIZZARDI, Giancarlo: On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta) Models. In: Proceedings of the 2007 conference on Databases and Information Sys- tems IV: DB&IS'2006. pp. 18–39. (2007).

GUIZZARDI, Giancarlo. Ontological foundations for structural conceptual models. 2005. Telematica Instituut / CTIT, *[S. l.]*, 2005.

ISO. ISO 9241-210:2019 - Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systemsInt. Organization for Standardization, 2019.

O'LEARY, Daniel E: Enterprise Knowledge Management. Computer 31(3), 54–61 (1998)

POLANYI, Michael. The Tacit Dimension. Garden City, NY: Doubleday, 1966.

PRESSMAN, R. S.; LOWE, D. Web engineering: a practitioner's approach. [S.l.]: McGraw-Hill Education, 2009. v. 1.

RIEHLE, D.: Framework Design: A Role Modeling Approach. Ph.D. Thesis, No. 13509. Zürich, Switzerland, ETH Zürich, 2000.

ROGERS, Yvonne, SHARP, Helen, PREECE, Jenny: Interaction Design: Beyond Human-Computer Interaction. Wiley Publishing, Chichester, United Kingdom, 3rd edn. (2011)

RUS, I., LINDVALL, M.: Knowledge management in software engineering. IEEE Soft- ware 19(3), 26–38 (2002)

RUY, Fabiano Borges.; FALBO, Ricardo de Almeida; BARCELLOS, Monalessa Perini; COSTA, Simone Dornelas; GUIZZARDI, Giancarlo. SEON: A Software Engineering Ontology Network. *In*: KNOWLEDGE ENG. AND KNOWLEDGE MANAGEMENT 2016, Anais [...]. : Springer, 2016. p. 527–542. Disponível em: <http://link.springer.com/10.1007/978-3-319-49004-5_34>.

SCHERP, Ansgar, SAATHOFF, Carsten, FRANZ, Thomas, STAAB, Steffen: Designing core ontologies. Applied Ontology. 6, 177–221 (2011).

SCHNEIDER, Kurt: Experience and Knowledge Management in Software Engineering. Springer Publishing Company, Incorporated, Heidelberg, Berlin, 1st edn. (2009)

STUDER, R., BENJAMINS, V.R., FENSEL, D.: Knowledge engineering: Principles and methods. Data & Knowledge Engineering. 25, 161–197 (1998).

SUÁREZ-FIGUEROA, Mari Carmen; GÓMEZ-PÉREZ, Asunción; MOTTA, Enrico; GANGEMI, Aldo.: Ontology Engineering in a Networked World. In: Ontology Engineering in a Networked World. 1–6. (2012).

SUTCLIFFE, Alistair G.: Requirements Engineering from an HCI Perspective. In: Soegaard, M., Dam, R.F. (eds.) The Ency. of Human-Computer Interaction, chap. 13, pp. 707–760. The Interaction Design Foundation, Aarhus, Denmark, 2nd edn. (2014)

VALASKI, Joselaine; MALUCELLI, Andreia; REINEHR, Sheila.: Review: Ontologies Application in Organizatinal Learning: A Literature Review. Expert Systems with Applications 39(8), 7555–7561 (2012)

VARMA, Vasudeva: Use of ontologies for organizational knowledge management and knowledge management systems. *In*: ontologies. [s.l.] : Springer, 2007. p. 21–47.