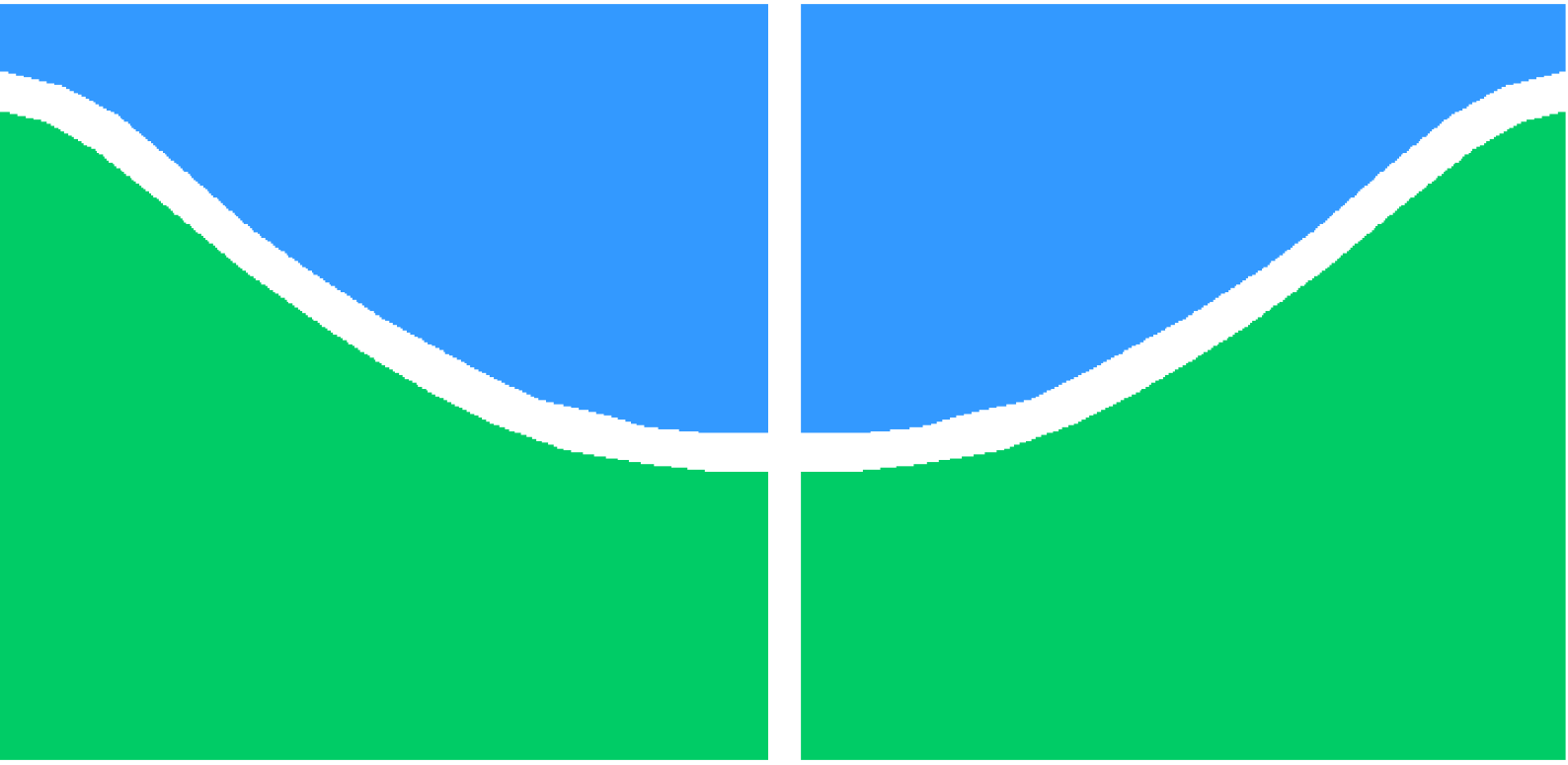


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Promoção da Qualidade de Vida no Trabalho no setor público: Proposta de uma ferramenta WEB para apoiar a coleta de dados do Inventário de Avaliação de Qualidade de Vida no Trabalho (IA_QVT)

Autor: Paulo Vítor Coelho da Rocha e João Gabriel Rossi

Orientador: Dra. Fabiana Freitas Mendes

Brasília, DF

2022

Paulo Vítor Coelho da Rocha e João Gabriel Rossi

Promoção da Qualidade de Vida no Trabalho no setor público: Proposta de uma ferramenta WEB para apoiar a coleta de dados do Inventário de Avaliação de Qualidade de Vida no Trabalho (IA_QVT)

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dra. Fabiana Freitas Mendes

Coorientador: Dr. Mário César Ferreira

Brasília, DF

2022

Paulo Vítor Coelho da Rocha e João Gabriel Rossi

Promoção da Qualidade de Vida no Trabalho no setor público: Proposta de uma ferramenta WEB para apoiar a coleta de dados do Inventário de Avaliação de Qualidade de Vida no Trabalho (IA_QVT)

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho apresentado. Brasília, DF, 26 de Setembro de 2022:

Dra. Fabiana Freitas Mendes
Orientador

Dr. Mário César Ferreira
Co-orientador

Prof. Dr. André Barros
Convidado 1

Dra. Letícia Alves Santos
Convidado 2

Brasília, DF
2022

Resumo

Contexto - O conceito de Qualidade de Vida no Trabalho (QVT) vem se tornando cada dia mais discutido dentro de organizações públicas e privadas. É de interesse das organizações poder medir o nível de QVT e para isso existe a abordagem denominada Ergonomia de Atividade Aplicada à Qualidade de Vida no Trabalho (EAA_QVT) que viabiliza essa mensuração por meio da aplicação do Inventário de Avaliação de Qualidade de Vida no Trabalho (IA_QVT). A aplicação dessa abordagem qualifica e quantifica o nível de QVT realizando uma pesquisa composta de questões abertas e fechadas. Para a aplicação desse método inovador de maneira otimizada é necessária a criação de uma nova plataforma *web*, pois a atual está desatualizada.
Objetivo - Este projeto tem como objetivo o desenvolvimento de uma plataforma *web* para a aplicação do IA_QVT e de artefatos necessários para sua concepção.
Método - A pesquisa foi iniciada com estudo e leituras sobre QVT e IA_QVT. Após o estudo foram utilizados quatro métodos, sendo eles a *Lean Inception* e *Product Backlog Building* (PBB), para o alinhamento da visão do produto, *Scrum*, para entregas contínuas, e *Kanban* para o desenvolvimento.
Resultados - Os métodos resultaram em artefatos utilizados para a documentação e desenvolvimento do produto de *software*. Os principais artefatos desenvolvidos foram o *backlog* do produto, o documento de arquitetura e o diagrama de *gantt*. Utilizando os artefatos, foi desenvolvido o produto viável mínimo (MVP).
Conclusões - Os métodos utilizados ajudaram o time a alinhar a visão do produto, escolher as tecnologias, construir o *backlog*, modelar o banco de dados e desenvolver o produto viável mínimo (MVP).

Palavras-chaves: Qualidade de vida no trabalho; inventário avaliativo; *web*; *software*.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IA_QVT	Inventário de Avaliação de Qualidade de Vida no Trabalho
CSV	*Comma-separated values*
EAA_QVT	Ergonomia da Atividade Aplicada a Qualidade de Vida no Trabalho
GT	Grupo de Trabalho
MVP	*Minimum viable product*
PBB	*Product backlog building*
PBI	*Product backlog items*
PcD	Pessoa com deficiência
QVT	Qualidade de vida no trabalho
TXT	*Text file*
US	História de usuário
UX	User Experience - Experiência do usuário
ZIP	Formato de compactação de arquivos
XLSX	Arquivo padrão do *Excel*

Sumário

1 Introdução

Muito se ouve de que as pessoas estão no núcleo central de prioridades das instituições públicas e empresas, porém, na prática, o discurso não poderia ser mais alheio à realidade. Uma mistura de cortes orçamentários, redução de equipes, terceirização, descontinuidade de projetos, entre outros fatores que colocam os indivíduos em segundo plano são práticas comuns. As consequências da priorização de resultados e a manutenção da ignorância do bem-estar dos trabalhadores trazem à tona uma realidade que mostra o adoecimento tanto físico quanto mental do trabalhador (FERREIRA, 2019).

Partindo da premissa do adoecimento físico e mental, pode-se observar que o surgimento de doenças dentro do ambiente de trabalho são causas reais de diminuição da produtividade e estão ligadas a baixa qualidade do ambiente de trabalho (HORST et al., 2018).

Nesse sentido, é fundamental falar de Qualidade de Vida no Trabalho (QVT), um termo criado com base no conceito do sistema sociotécnico. O sistema sociotécnico tem como objetivo analisar a autonomia no trabalho, interdependência e alto-envolvimento com a ideia de melhor encaixe entre tecnologia e organizações sociais (HORST et al., 2018).

Para FERREIRA (2019), o contexto de trabalho é a principal variável de ajuste no que diz respeito à QVT, este permeia todos os integrantes de uma organização na busca pelo alinhamento do bem-estar do indivíduo e da produtividade organizacional sustentável.

Nessa perspectiva, a QVT é definida sob duas óticas interdependentes. Na ótica das organizações é um preceito de gestão organizacional que se expressa por um conjunto de normas, diretrizes e práticas no âmbito das condições, da organização e das relações socioprofissionais de trabalho que visa a promoção do bem-estar individual e coletivo, o desenvolvimento pessoal dos trabalhadores e o exercício da cidadania organizacional nos ambientes de trabalho.

Sob a ótica dos trabalhadores, ela se expressa por meio das representações globais (contexto organizacional) e específicas (situações de trabalho) que estes constroem, indicando o predomínio de experiências de bem-estar no trabalho, de reconhecimentos institucional e coletivo, de possibilidade de crescimento profissional e de respeito às características individuais FERREIRA (2017).

Para se promover QVT de acordo com os fundamentos apresentados no conceito de FERREIRA (2019), torna-se necessário mensurar a QVT e analisá-la quantitativa

e qualitativamente com rigor científico. Essa mensuração é viabilizada pela aplicação da abordagem intitulada Ergonomia da Atividade Aplicada à Qualidade de Vida no Trabalho (EAA_QVT) que utiliza na fase de diagnóstico de QVT o Inventário de Avaliação da Qualidade de Vida no Trabalho (IA_QVT), que pode ser realizado no formato impresso ou digital.

A plataforma atualmente para aplicação digital está desatualizada, portanto, é necessária a criação de uma nova ferramenta. Essa ferramenta deve ter como foco a coleta, o armazenamento, o tratamento e a visualização dos resultados das aplicações do IA_QVT, facilitando assim a aplicação da EAA_QVT nas instituições públicas e contribuindo para promoção do bem-estar no trabalho.

O propósito deste documento é explicitar o processo do desenvolvimento de um produto de *software* que permita realizar um diagnóstico rápido e com rigor científico, de como os trabalhadores avaliam a QVT na organização em que trabalham e gerar subsídios fundamentais para a concepção de uma política ou Programa de Qualidade de Vida no Trabalho (PQVT) com base na realidade apontada pelos trabalhadores.

1.1 Objetivos

Este trabalho tem como objetivo principal desenvolver um sistema que coleta, armazena, trata e disponibiliza dados sobre qualidade de vida no trabalho. Para alcançar esse objetivo foram definidos os seguintes objetivos específicos:

- **OE1**: Identificar requisitos para coleta, tratamento, armazenamento e visualização de dados.
- **OE2**: Desenvolver uma solução capaz de coletar, tratar, armazenar e visualizar os dados da aplicação.
- **OE3**: Desenvolver um sistema que garanta a confidencialidade das informações dos usuários durante a etapa da coleta de dados.

Com isso foram definidos o contexto, o problema, a justificativa e os objetivos do trabalho. Para prosseguir, no próximo capítulo serão apresentados e definidos os conceitos importantes para o entendimento do projeto. Também serão apresentadas as tecnologias utilizadas para o desenvolvimento do produto de *software*.

2 Referencial teórico

Este capítulo tem por finalidade definir e detalhar os principais termos necessários para o entendimento do projeto. Serão abordados conceitos sobre QVT, métodos de desenvolvimento, ciclo de vida e qualidade de software. Também será abordado as ferramentas utilizadas para suporte tecnológico.

2.1 Qualidade de Vida no Trabalho

Qualidade de vida no trabalho é um conceito datado dos anos 70, mas que carrega um história datada do século passado. O conceito tem seu início em um período de avanço para o trabalhador, época em que legislações trabalhistas foram criadas proibindo o trabalho infantil e estabelecendo cargas horárias máximas de trabalho (WALTON, 1973).

Para WALTON (1973), o conceito carrega toda a história de movimentos de reforma dos anos 30, que tiveram ênfase em remuneração e segurança no ambiente de trabalho. Com a modernização do trabalho, o conceito também se modernizou, principalmente por sua ligação com produtividade, qualidade tão desejada pelas empresas.

WALTON (1973) separou em oito critérios, criando uma estrutura para a análise do conceito, sendo eles:

1. Compensação justa e adequada: o trabalhador utiliza do trabalho como maneira de se sustentar, então o quão bem esse conceito é avaliado afeta diretamente a qualidade de vida no trabalho.

2. Condições de saúde e segurança no ambiente de trabalho: os trabalhadores não devem ser expostos à condições que trazem prejuízos a sua saúde, seja em relação a local de trabalho ou carga horária.

3. Oportunidade de usar e avançar as capacidades humanas: dividida em cinco grandes partes, sendo elas: autonomia, quantidade de habilidades utilizadas no trabalho, informação e perspectiva do processo de trabalho e resultado das ações, participação no planejamento de atividades e divisão de tarefas.

4. Oportunidades futuras para crescimento contínuo e segurança: pode ser dividida em quatro partes, sendo elas (1) desenvolvimento, (2) oportunidades de utilizar conceitos e habilidades aprendidas em momentos futuros no trabalho, (3) oportunidades de avanço na carreira e (4) segurança financeira e de trabalho.

5. Integração social no ambiente de trabalho: o ambiente de trabalho normalmente é tratado tal qual uma organização social, então, as relações sociais devem ser consideradas como parte importante da qualidade de vida no trabalho.

6. Constitucionalismo na organização de trabalho: o trabalhador é afetado diretamente por decisões e ações de seus empregadores, portanto possui direitos que devem ser garantidos no ambiente de trabalho, sendo os principais: privacidade, equidade e liberdade de expressão.

7. O trabalho e o espaço total da vida: olhando para o indivíduo trabalhador e suas experiências no ambiente de trabalho, é perceptível o impacto em outras áreas da sua vida, principalmente em relação à carga horária e ao tempo para cultivo da vida privada. Portanto, é necessário achar um equilíbrio entre as diferentes esferas da vida.

8. Relevância social da vida no trabalho: qual o impacto social que a organização do trabalho exerce na vida do trabalhador? Como ele classifica e enxerga as ações da empresa da perspectiva social? Organizações irresponsáveis afetam a perspectiva do trabalhador negativamente.

Levando em consideração os critérios citados anteriormente o modelo descritivo de Ergonomia da Atividade Aplicada a Qualidade de Vida no Trabalho (EAA_QVT), analisa a qualidade de vida no trabalho sob um grupo de variáveis, sendo elas: condições de trabalho, organização do trabalho, relações socioprofissionais, reconhecimento e crescimento profissional e elo trabalho-vida social (FERREIRA, 2011).

Pode ser observado que o modelo descritivo teórico-metodológico da EAA_QVT e os oito critérios seguem a mesma linha para definição e validação do conceito. Com isso, pode ser entendido que o conceito vem evoluindo por mais de 50 anos e o porquê de ser importante no século 21.

Para a existência da produtividade saudável dentro do ambiente de trabalho, o fator qualidade de vida no trabalho não pode ser desconsiderado. O trabalhador é quem produz e é para ele que o ambiente de trabalho deve ser moldado, não o contrário.

Por fim para mensurar a QVT é utilizado o instrumento de pesquisa chamado IA_QVT (Inventário de Avaliação de Qualidade de Vida no Trabalho) que é um instrumento que permite conhecer com rigor científico o que pensam os respondentes sobre QVT.

O conceito de Qualidade de Vida no Trabalho foi definido e explicado sob as diferentes óticas e para entender como será desenvolvido o *software*, será discutido a seguir os métodos para o desenvolvimento de software.

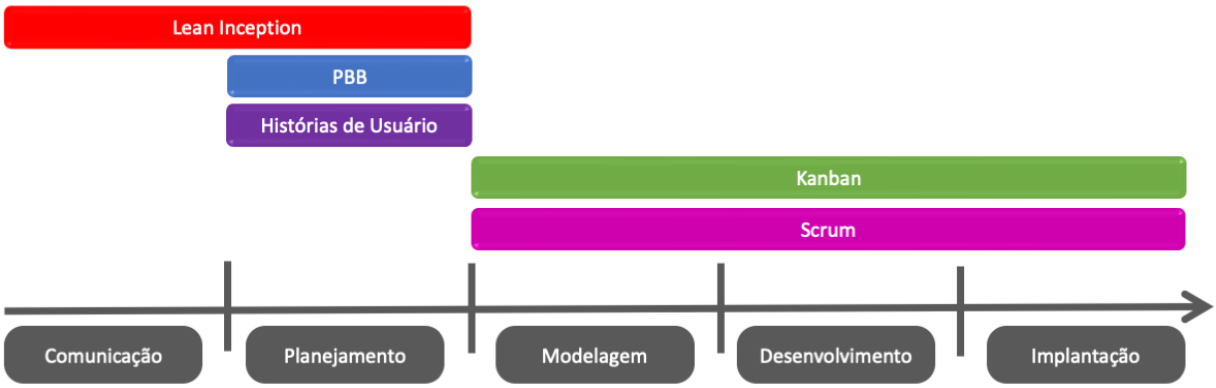
2.2 Métodos de Desenvolvimento de Software

O conceito de QVT é antigo e ao mesmo tempo atual. Mas o que isso significa? Como explicado na Seção 2.1, o conceito está em constante processo de evolução. Com isso, as metodologias baseadas nele necessitam estar se atualizando, seja em conteúdo, ou em maneiras de aplicação.

A proposta de inovação em questão visa possibilitar que a metodologia seja aplicada de maneira digital. A facilidade de gerar relatórios, armazenar dados e controlar acessos permite que o pesquisador otimize o tempo da aplicação da metodologia. Por conta da necessidade de inovação, nasce esse produto de *software*. Ele será desenvolvido para a aplicação de metodologias de QVT em ambientes de trabalho diversos.

Para o desenvolvimento do produto serão utilizados cinco métodos diferentes. Cada um deles está conectado a uma atividade do modelo de processos escolhido pela equipe, veja na Figura 1.

Figura 1 – Modelo de ciclo de vida do produto



Fonte: Os próprios autores (2022e)

Com isso, cada atividade do modelo de ciclo de vida possui ao menos um método relacionado a si.

2.2.1 *Lean Inception*

De acordo com CAROLLI (2018), *Lean Inception* é um método de alinhamento de pessoas sobre o produto mínimo viável (*Minimun Viable Product*, MVP) a ser construído. O produto mínimo viável é uma versão do produto que possui um conjunto mínimo de funcionalidades. Este visa satisfazer as necessidades do cliente e, assim, coletar o *feedback*.

Esse método é utilizado nas atividades de comunicação e planejamento do modelo de processos, vide Seção 2.3. A atividade de comunicação é utilizada para alinhar a visão do produto e conhecer o usuário. A atividade de planejamento é utilizada para descrever as

funcionalidades pela primeira vez, criar a primeira versão do cronograma com as devidas priorizações e planejar o produto mínimo viável (MVP).

A *Lean Inception* é um método enxuto, que leva no máximo uma semana se realizada da maneira recomendada. Ao finalizá-la, o time possui a visão do produto alinhada e o planejamento do MVP ao invés de histórias de usuário pontuadas (CAROLLI, 2018).

A *Lean Inception* possui nove etapas, as quais são realizadas de maneira sequencial, sendo elas:

- Visão do Produto: etapa em que é explicitada a visão do produto de cada participante (CAROLLI, 2018).
- Produto: etapa em que cada participante escreve o que o produto é, não é e o que faz e não faz (CAROLLI, 2018).
- Objetivos do Produto: cada participante escreve três objetivos de negócio do produto. A partir disso, a equipe os divide em no mínimo cinco categorias (CAROLLI, 2018).
- Personas: criam-se três perfis para possíveis usuários da aplicação com o intuito de entender-los e prepará-los para a próxima etapa (CAROLLI, 2018).
- Jornada do Usuário: utilizando as personas criadas anteriormente, os participantes escrevem um objetivo para a persona e criam a sua jornada, desde o início até chegar ao objetivo, que é relacionado ao uso do produto (CAROLLI, 2018).
- *Brainstorming* de Funcionalidades: cada participante preenche três funcionalidades e equipe as divide em categorias (CAROLLI, 2018).
- Revisão Técnica, de Negócio e UX: utilizando as funcionalidades criadas no passo anterior, cada participante seleciona uma destas e a classifica de acordo com as capacidades técnicas da equipe. Atribui-se um valor de um a três para o quanto esta funcionalidade vai demandar nos atributos: quantidade de esforço, valor de negócio e UX (CAROLLI, 2018).
- Sequenciador: as funcionalidades são classificadas, pela equipe, por *sprints*. Determina-se quando será implementada cada funcionalidade, o MVP e seus incrementos (CAROLLI, 2018).
- Canvas MVP: é o artefato final da *Lean Inception*. Ele resume os passos anteriores e estabelece as métricas de validação do produto (CAROLLI, 2018).

Ao finalizar as nove etapas, a equipe pode optar por utilizar métodos adicionais para alcançar um maior grau de detalhamento dos requisitos ou partir diretamente para a escrita do *backlog*.

2.2.2 *Product Backlog Building*

Product Backlog Building (PBB) é um método de detalhamento das funcionalidades do produto de *software*. Isso facilita a criação do *backlog* do produto.

O PBB é um método que pode utilizar artefatos gerados durante a *Lean Inception* para otimizar o seu tempo de aplicação, mas também pode ser feito de maneira independente.

O método utiliza o *canvas* PBB, um *template* dividido em cinco etapas. As quatro primeiras etapas dele existem para facilitar a criação do artefato *product backlog items* (PBI). De acordo com CAROLLI (2019), as cinco etapas presentes no PBB são:

- Problemas: identifica as dificuldades do cliente.
- Expectativas: identifica os anseios do cliente.
- Personas: identifica os usuários finais do produto, como eles estão envolvidos, o que realizam e o que esperam.
- *Features*: identifica as funções que cada persona vai realizar no produto final, o que necessitam para realizá-las e o que esperam com sua utilização.
- *Product Backlog Item*: divide as *features* em funcionalidades menores e facilita a construção final do *backlog* e das histórias de usuário.

Após finalizar essas cinco etapas, o método é encerrado. Recomenda-se transcrever o PBI no formato de histórias de usuário, utilizando as *features* como base para os épicos.

2.2.3 Histórias de Usuário e Épicos

Histórias de usuário são descrições informais em linguagem natural das funcionalidades, baseando-se na perspectiva do usuário. Elas descrevem o tipo de usuário que vai utilizar a funcionalidade, o que ele deseja da funcionalidade e o porquê de sua necessidade. (PARADIGM, 2022b).

O formato de histórias de usuário é uma maneira de escrever o *backlog*. Ela consiste de épicos (macro-tarefas) que descrevem de maneira geral as funcionalidades do produto e histórias de usuário (micro-tarefas) agrupadas dentro de épicos.

Por fim, são atribuídas pontuações às histórias de usuário. Existem diferentes maneiras de pontuá-las, mas a mais comum é utilizando a sequência de *fibonacci*, pois a mesma cresce proporcionalmente a cada adição de valores, facilitando assim a visualizar e entender a diferença de cada história pontuada (COHN, 2019). Para a pontuação específica de cada história, uma história base é selecionada e, a partir dela, as outras são pontuadas.

A pontuação é baseada em tempo, risco, complexidade e valor de negócio (PARADIGM, 2022a). Após sua conclusão, é encerrada a fase de planejamento e iniciada a fase de modelagem.

2.2.4 *Scrum*

O *Scrum* é um método ágil que oferece às organizações de software um novo e dinâmico modo de gestão de equipes. O método tem sido bastante aplicado ao desenvolvimento de software (KARABULUT; ERGUN, 2014).

Ele é baseado na ideia de que muitos processos no desenvolvimento de software são difíceis de serem previstos. Sendo assim, o *scrum* facilita a flexibilização no desenvolvimento de software (KARABULUT; ERGUN, 2014). O objetivo desse método é entregar software de alta qualidade após as *sprints* - um conjunto de períodos definidos de tempo.

Os quatro elementos de um *scrum* são uma biblioteca de tarefas, uma equipe, uma *sprint* e uma reunião do *scrum*. Primeiramente, as histórias de usuário são definidas como itens de trabalho e, posteriormente, são inscritas na biblioteca de tarefas. Em seguida, são atribuídos aos membros da equipe por meio da reunião realizada no estágio inicial da *sprint* - denominado *sprint planning* (planejamento da *sprint*) (KARABULUT; ERGUN, 2014).

Durante os dias da semana são realizados encontros diários com duração máxima de 15 minutos onde cada membro deve responder três perguntas: “O que eu fiz ontem?”, “O que farei hoje?” e “Quais impedimentos estão no meu caminho?” (KARABULUT; ERGUN, 2014)

O método *Scrum* é utilizado durante as atividades de modelagem, desenvolvimento e implantação. Nas três etapas ele é utilizado para controlar o andamento das tarefas propostas, além de ajudar os desenvolvedores a alinhar o trabalho com o restante do time durante os encontros propostos.

A primeira etapa é o planejamento da *sprint*. Ela acontece no início de cada *sprint* e tem como objetivo montar o *Sprint backlog*, que contém uma lista de tarefas a serem realizadas e a pontuação de cada uma. A segunda etapa é a revisão, utilizada para o controle das tarefas separadas para a *sprint*, na qual as mesmas são validadas e os *feedbacks* da equipe são coletados. Por fim, a terceira etapa, chamada de retrospectiva, é uma cerimônia que deve acontecer sempre no último dia da *sprint*. Nesta cerimônia o objetivo é analisar tudo o que aconteceu durante a *sprint* e propor melhorias para a próxima.

2.2.5 *Kanban*

A abordagem do *Kanban* foi introduzida na indústria manufatureira japonesa na década de 1950. *Kanban* é uma palavra japonesa que significa cartaz, e é usado nas

fábricas como um sistema de controle de fluxo na produção *Just In Time*. A ideia por trás do *Kanban* é executar o pensamento enxuto na prática (AHMAD; MARKKULA; OIVO, 2013).

O método *Kanban* no desenvolvimento de *software* permite que as equipes visualizem o fluxo de trabalho, definam o trabalho em progresso em cada estágio e mensurem o ciclo de tempo de cada um desses fluxos.

O método utiliza um quadro chamado quadro *Kanban*, que é dividido em diferentes etapas para acompanhar o progresso das tarefas. O quadro *Kanban* atribui transparência ao processo de desenvolvimento de *software*, pois mostra as tarefas de trabalho de cada um dos desenvolvedores. Com isso, o quadro permite uma comunicação mais clara e rápida identificação de gargalos.

Outro de seus objetivos é minimizar o trabalho em progresso, desenvolvendo apenas o que foi pedido naquele momento. Essa minimização permite a criação de um fluxo constante de entregas ao cliente, já que os desenvolvedores focam apenas em algumas tarefas e não todas (AHMAD; MARKKULA; OIVO, 2013).

O *Kanban* é comumente utilizado durante as atividades de desenvolvimento e implantação, mas pode ser útil durante a modelagem. Nas atividades de desenvolvimento e implantação, é utilizado para o controle do desenvolvimento das histórias de usuário. Já durante a modelagem é utilizado para o controle de artefatos criados durante a atividade, como o documento de arquitetura e a modelagem do banco.

2.3 Ciclo de vida de um *software*

O desenvolvimento do produto de *software* passa por diversas fases, desde o entendimento do produto até a entrega e manutenção. Esse conjunto de fases é conhecido como o ciclo de vida do *software*. Para falar de ciclo de vida, é necessário primeiro entender o conceito de processo e atividade no contexto de *software*.

Para Sommerville (2011), uma atividade é uma tarefa que possui um resultado esperado, como fazer o desenho de uma interface de usuário ou modelar um banco de dados. Já um processo de *software* é um conjunto de atividades correlacionadas que possuem como resultado uma funcionalidade ou artefato relacionados ao produto final de *software*.

Após entender o que é um processo, o conceito de um modelo de ciclo de vida fica mais simples de entender. Modelo de ciclo de vida nada mais é do que a separação de atividades de certos processos para guiar o time durante a criação do produto.

Para Pressman e Maxim (2015), um *framework* genérico de modelo de ciclo de vida possui cinco atividades, sendo elas: comunicação, planejamento, modelagem, desenvolvimento e implantação conforme ilustra a Figura 1.

As atividades de comunicação, planejamento e modelagem caminham juntas, pois são realizadas antes da escrita do código. A atividade de comunicação tem como foco entender o produto e as suas nuances, também conhecido como visão do produto. No fim dessa atividade o time cria um esboço das funcionalidades do produto (PRESSMAN; MAXIM, 2015).

O planejamento utiliza o esboço das funcionalidades criado na *Lean Inception*, referenciada na Seção 2.2.1, para definir cronogramas, riscos e custos. A partir deste esboço, as funcionalidades são descritas de maneira mais técnica, por meio das histórias de usuário que são definidas e explicadas na Seção 2.2.3 deste documento (PRESSMAN; MAXIM, 2015).

Já a modelagem é uma atividade mais técnica que consiste em documentar como o *software* será organizado internamente. Nessa atividade é definida a arquitetura do produto, a organização dos dados dentro do banco de dados e outras partes que compõe o produto (PRESSMAN; MAXIM, 2015).

Por fim, realiza-se as atividades de desenvolvimento e implantação. A atividade de desenvolvimento é composta pela escrita do código, com base na documentação gerada nas três atividades anteriores. Já a implantação é a liberação do produto de *software* para o consumidor utilizar e avaliar (PRESSMAN; MAXIM, 2015). É importante ressaltar que para a realização dessas duas etapas é importante possuir conhecimentos sobre qualidade de *software*, tópico que será abordado na próxima seção.

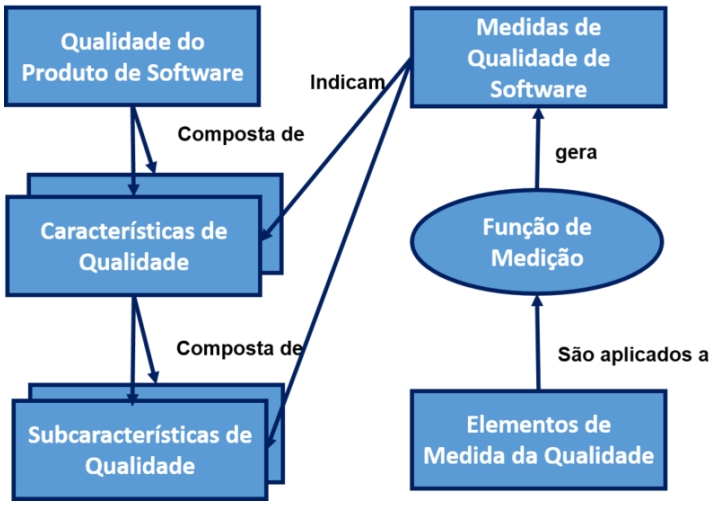
2.4 Qualidade de *software*

Para Pressman e Maxim (2015) a qualidade de *software* pode ser definida por: “um processo de *software* eficaz aplicado de forma a criar um produto útil, que forneça valor mensurável para quem o produz e para quem o usa”. Essa definição serve para enfatizar três pontos principais:

- Um processo de *software* eficaz estabelece uma infraestrutura capaz de suportar a construção de um produto de *software* de alta qualidade. Seu correto gerenciamento reduz aspectos negativos e ajuda a evitar o caos do projeto – um fator chave para a má qualidade.
- Um produto útil oferece o conteúdo, as funções e os recursos que o usuário final deseja, mas também entrega esses ativos de maneira confiável e livre de erros.
- Ao agregar valor, tanto para o produtor quanto para o usuário de um produto de *software*, o *software* de alta qualidade oferece benefícios para a sua organização, tendo em vista um menor esforço de manutenção e menor necessidade de correções de *bugs* para a comunidade que recebe um produto que satisfaz suas necessidades.

A norma ISO/IEC 25020 (2019) propõe um modelo de referência apresentado na Figura 2 para a medição de qualidade de produto de *software*. Este modelo mostra que a qualidade do produto é composta de características e subcaracterísticas de qualidade.

Figura 2 – Modelo de Referência de Qualidade



Fonte: (ISO/IEC 25020, 2019)

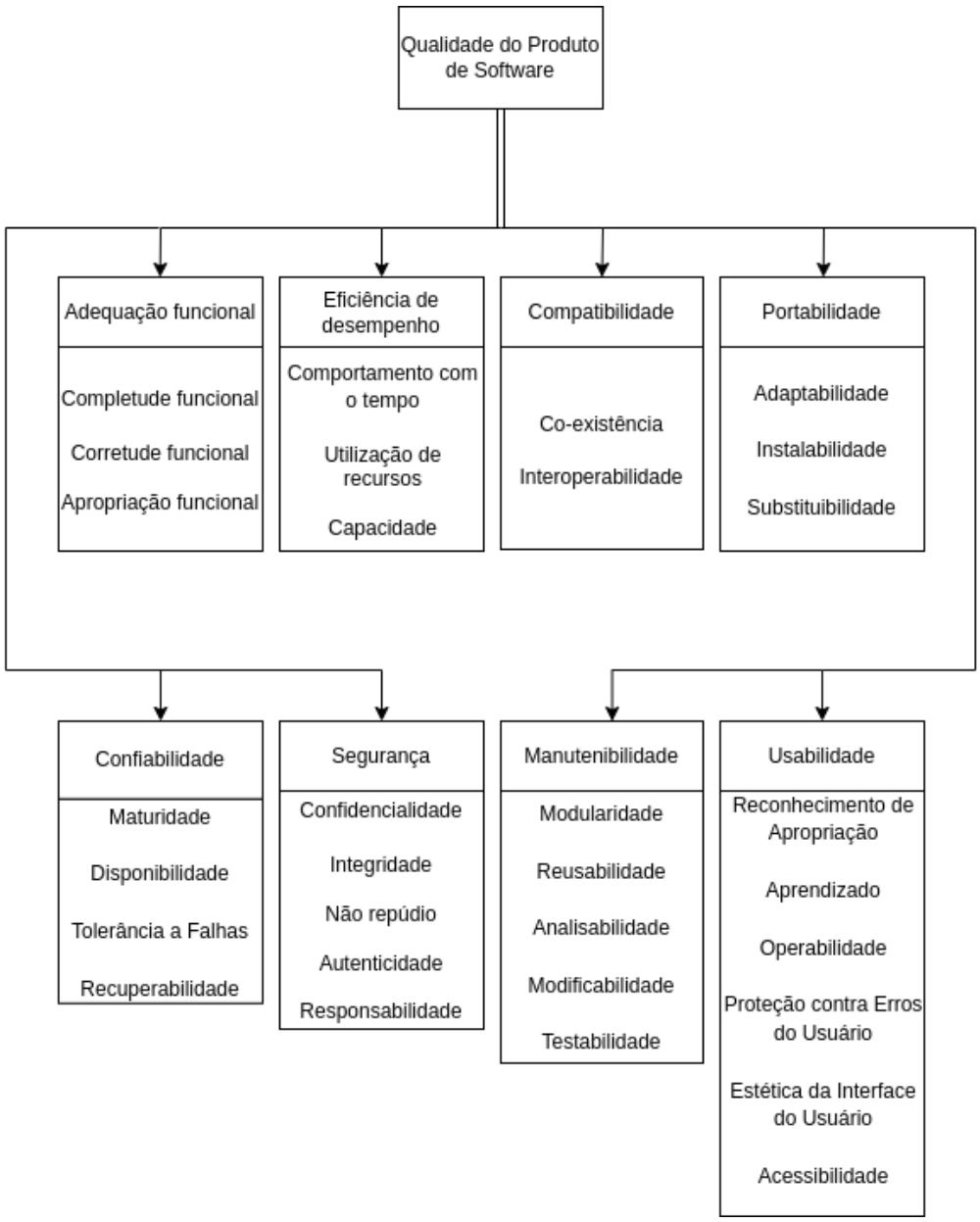
A norma ISO/IEC 25010 (2011) divide o conceito de qualidade em três aspectos diferentes: qualidade de processo, qualidade de produto e qualidade em uso.

A qualidade de processo é definida e especificada na norma ISO/IEC 15504 sendo apenas citada na norma ISO/IEC 25010:2011.

A qualidade em uso é definida por ser a capacidade do produto de *software* de permitir que usuários específicos atinjam metas com eficácia, produtividade, segurança e satisfação em contextos de uso especificados.

A qualidade de produto é dividida em oito características (adequação funcional, confiabilidade, eficiência de desempenho, usabilidade, segurança, compatibilidade, manutenibilidade e portabilidade), sendo que cada uma das características é composta por um conjunto de subcaracterísticas relacionadas como mostra a Figura 3.

Figura 3 – Características e subcaracterísticas de qualidade



Nota: Adaptação própria (ISO/IEC 25010, 2011)

As características definidas por ambos os modelos são relevantes para todos os produtos de software e sistemas computacionais. Elas fornecem uma terminologia consistente para especificar, medir e avaliar a qualidade do sistema e do produto de software. Além disso, também fornecem um conjunto de características de qualidade em que os requisitos de qualidade declarados podem ser comparados à sua completude.

2.5 Suporte tecnológico

Nesta seção serão apresentados os instrumentos e ferramentas utilizadas pela equipe para o desenvolvimento do *software*.

2.5.1 *Backend*

O *backend* é toda a parte da programação voltada ao funcionamento interno de um software. Nele estão presentes linguagens, *frameworks* e tecnologias que serão utilizados nos microsserviços do *software*. O detalhamento sobre o *backend* pode ser encontrado no Apêndice A onde está o documento de arquitetura.

Foi utilizada como linguagem de desenvolvimento do *backend* o *Python* que é uma linguagem de programação de alto nível, interpretada e multi-paradigma, normalmente utilizada junto com o paradigma de orientação à objetos. Por ser uma linguagem de fácil interpretação, e de sintaxe simples e direta, são utilizadas poucas linhas para execução de tarefas complexas (PYTHON, 2022).

O *framework* escolhido para compor o *backend* foi o *Django REST Framework* por conta da sua flexibilidade para construção de *APIs* e da familiaridade da equipe com ele, o que diminui os riscos do desenvolvimento.

Por fim, o banco de dados escolhido para compor os microsserviços foi o PostgreSQL. Trata-se de um sistema de banco de dados relacional de objeto de código aberto com mais de 30 anos de desenvolvimento ativo que lhe rendeu uma forte reputação de confiabilidade, robustez de recursos e desempenho (POSTGRESQL, 2022).

2.5.2 *Frontend*

Pode ser classificado como a parte visual de um site, aquilo que é passível de interação direta com o usuário. Nesta seção será explicitado as linguagens, *frameworks* e tecnologias que serão utilizados no *frontend* do *software*.

Foi utilizada como linguagem de desenvolviemnto do *frontend* o *Javascript* que é uma linguagem de programação interpretada, multiparadigma e com funções de primeira classe. É uma linguagem coringa, pois é utilizada como uma linguagem padrão para *frontend*, mas pode ser utilizada de diversas outras maneiras, desde aplicações *desktop* até o *backend* (CONTRIBUTORS, 2021).

O *framework* escolhido para compor *frontend* foi o *ReactJS* que utiliza componentes para a criação de interfaces de usuário interativas, facilitando o desenvolvimento de um sistema complexo, criando vários componentes simples e os combinando (SOURCE, 2022).

2.5.3 *Deploy*

O *deploy* é o processo pelo qual o *software* passa para ser colocado em produção. Foi utilizado o *Docker* como ferramenta de virtualização de sistemas operacionais para a execução do código. Ele permite a aquisição de todas as dependências que um *software* necessita para funcionar. Virtualizar uma máquina ajuda a evitar que ocorram problemas de conflito entre as versões, seja em produção, homologação ou localmente (DOCKER, 2022b).

Para tornar possível a comunicação entre os ambientes virtualizados, foi utilizado o *docker-compose*. Ele foi criado para facilitar a conteinerização de aplicações em que existem contêineres com mais de 2 imagens, por exemplo, executar pelo mesmo *script* um serviço de *backend* juntamente com seu banco de dados (DOCKER, 2022a).

2.5.4 *Gerenciamento*

Nesse tópico estão apresentadas as ferramentas para o controle de versão, gerenciamento do código e aplicação de métodos. Nesse contexto, foi utilizado o *Github* que é uma ferramenta de controle de versão em código aberto que foi criada para rodar diretamente em *kernels linux*, portanto, é extremamente rápida e eficaz. Ela possui um sistema de *branch*, que permite a criação de ramificações diferentes e independentes, que podem ser unidas ao realizar as ações de *merge*, facilitando assim o desenvolvimento paralelo (GIT, 2022).

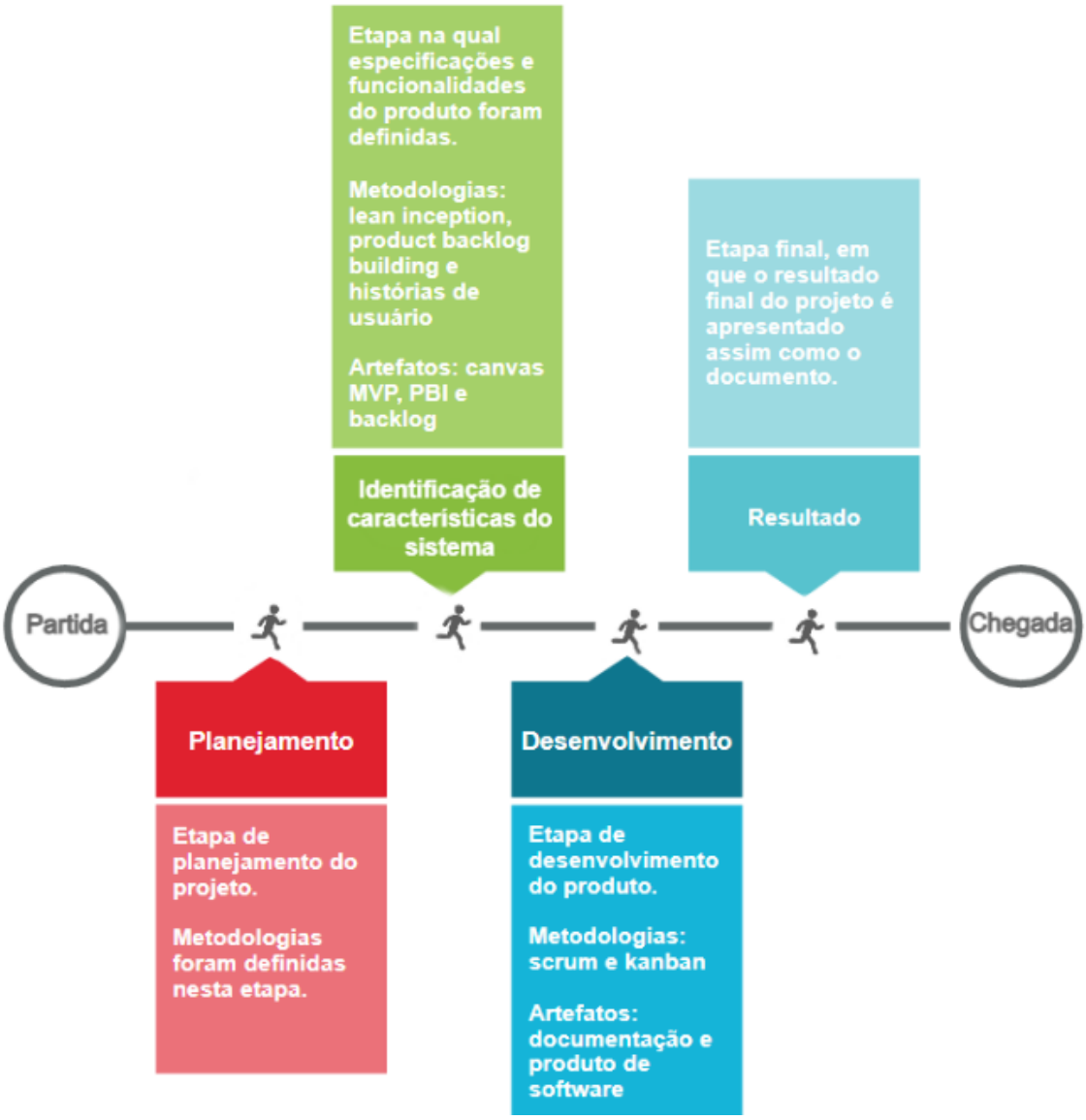
Por fim, foi utilizado o *Zenhub* que é uma ferramenta de gerenciamento de projeto colaborativa criada primariamente para ser utilizada com o *github*. A ferramenta possui integração direta do quadro *kanban* com os repositórios e as suas *issues*, além da criação automática de gráficos sobre o trabalho da equipe utilizados no *scrum* (ZENHUB, 2022).

Após descritos os métodos de desenvolvimento de *software* e as tecnologias que foram utilizadas pela equipe, será explicado no capítulo seguinte, como foi realizada a aplicação desses métodos no desenvolvimento do produto.

3 Métodos de Apoio

A execução do trabalho foi dividida em quatro etapas, sendo elas: (1) planejamento; (2) entendimento; (3) desenvolvimento da solução e (4) síntese e apresentação. Cada etapa conta com suas nuances, artefatos e métodos, como pode ser visto na Figura 4.

Figura 4 – Diagrama de etapas dos métodos



Fonte: Os próprios autores (2022b)

A Figura 4 possui o diagrama de etapas dos métodos. Este diagrama contém uma linha sequencial lógica de cada etapa do projeto, além de sua breve descrição, explicando as técnicas e métodos empregados para sua execução. As seções a seguir explicitam cada uma dessas etapas, com seus métodos e artefatos.

3.1 Planejamento

Na etapa de planejamento, o tema foi apresentado pelos orientadores. Também foi realizado um estudo mais profundo sobre QVT e IA_QVT com o objetivo de entender mais sobre esse conceito e sua aplicação.

Durante esse estudo, o contexto, o problema e a justificativa do trabalho foram descritos para nortear o desenvolvimento do projeto. Por fim, foi criado o cronograma geral do projeto e os métodos da próxima etapa foram definidos.

3.2 Entendimento e *Backlog*

A etapa de entendimento possui duas fases. A primeira fase consiste em alinhar a visão do time em relação ao produto, começar a descrever as funcionalidades e criar um cronograma básico. Já a segunda fase adicionará mais detalhes nas funcionalidades do produto.

Para a primeira fase foi utilizado o método *Lean Inception*. Posteriormente, utilizando os artefatos gerados foi realizada a segunda fase, o PBB.

3.2.1 *Lean Inception*

A *Lean Inception* é útil quando realizada em conjunto com o cliente, seja em todas as sessões ou apenas na validação. Já que a visão do cliente sobre os artefatos é importante para o entendimento do produto por parte do time. Por isso, o time optou por realizar o *workshop* em conjunto com o cliente, ao definir um horário para a participação de todos os membros.

Neste projeto, a sincronização de horários se apresentou como uma dificuldade, pois a *Lean Inception* recomenda que os encontros sejam realizados em dias inteiros durante uma semana, o que era inviável para o time. Portanto foi necessário alterar a aplicação do método em sessões de uma hora, normalmente realizadas uma vez por semana.

Para isso, foi utilizada uma mesclagem de atividades síncronas e assíncronas. As atividades síncronas foram utilizadas para discussões e criações de alguns artefatos. Já nas assíncronas, cada um dos participantes preenchia algum artefato escolhido durante as reuniões síncronas, como uma tarefa de casa. O cronograma de execução do método pode ser visto na Tabela 1.

Tabela 1 – Cronograma *Lean Inception*

Etapa da *Lean Inception*	**Semana**				
	23 a 29/01	30/01 a 05/02 a	06 a 12/02	13 a 19/02 a	20 a 26/02
Visão do Produto	X				
O Produto	X				
Objetivos do Produto	X	X			
Personas		X	X		
Jornada do Usuário		X	X		
Brainstorming de Funcionalidades			X	X	
Revisão técnica, de negócio e UX				X	X
Sequenciador				X	X
Canvas MVP					X

Fonte: Os próprios autores (2022e)

As etapas mostradas na Tabela 1 foram realizadas pelo time da seguinte maneira:

- Visão do produto: realizadas de maneira assíncrona, cada participante preencheu o artefato durante a semana e em uma reunião síncrona posterior o artefato foi montado.
- Objetivos do produto: separado em duas partes, uma atividade assíncrona para os membros da equipe e uma síncrona, na qual foram feitos os agrupamentos dos objetivos.
- Personas e Jornada de Usuário: realizada também em duas partes, uma assíncrona, onde foram preenchidos os artefatos, e uma síncrona, onde foram apresentados pelo time.
- *Brainstorming* de Funcionalidades: tal qual os objetivos, o preenchimento foi feito de maneira assíncrona e o agrupamento de maneira síncrona.
- Revisão técnica, de negócio e UX: realizada apenas pelo time de desenvolvimento, foi feita em apenas uma reunião, validada pelos professores de maneira assíncrona e discutida em reunião posterior.
- Sequenciador: Feito novamente apenas pelo time de desenvolvimento em reunião e validado pelos professores, foi discutido na mesma reunião que o artefato anterior.
- *Canvas MVP*: feito pelo time de desenvolvimento de maneira síncrona e validado pelos professores.

A *Lean Inception* durou cinco semanas, começando dia 23 de janeiro e encerrada no dia 26 de fevereiro. Foram criados nove artefatos durante as cinco semanas. Com isso, o time concluiu a aplicação do método e pode seguir para a próxima fase, o PBB.

3.2.2 PBB

O PBB não é um método que se beneficia muito da presença do cliente quando feito em conjunto com a *Lean Inception*, pois o entendimento do produto e as funcionalidades já foram explicitados, como pode ser visto na Seção 2.2.2. Portanto, foi realizado apenas com o time de desenvolvimento.

Pelo fato do time de desenvolvimento possuir o horário mais flexível e a metodologia possuir menos etapas, foi necessário um encontro síncrono para sua conclusão. Além desse encontro, um artefato foi feito de maneira assíncrona. Na Tabela 2 pode ser visto o cronograma do método.

Tabela 2 – Cronograma PBB

Etapas PBB	Semanas	
	06 a 12/03	13 a 19/03
Problems	X	
Expectations	X	
Personas	X	
Features	X	
Project Backlog Items		X

Fonte: Os próprios autores (2022e)

As etapas referenciadas na Tabela 2 foram realizadas pelo time da seguinte maneira:

- *Problems*: realizada de maneira síncrona pela equipe de desenvolvimento, essa etapa foi feita baseada na *Lean Inception*.
- *Expectations*: realizada durante a mesma reunião da etapa de *problems* e feita da mesma forma.
- Personas: realizada na mesma reunião e baseada nas personas da *Lean Inception*, sofreu algumas alterações para se adequar ao PBB.
- *Features*: realizada em conjunto com as últimas três etapas, foi baseada nas *features* do artefato final da *Lean Inception*.
- *Project Backlog Items*: feito com as *features* da *Lean Inception* de maneira assíncrona.

O PBB começou dia 6 de março e foi finalizado no dia 19 do mesmo mês. Foram criados cinco artefatos, quatro em encontro síncrono e um de maneira assíncrona. Com isso, o PBB foi finalizado e o time pode avançar para as próximas etapas.

3.3 Desenvolvimento da solução

A etapa de desenvolvimento da solução utiliza dois métodos complementares. O primeiro método tem como objetivo o gerenciamento do time. O segundo tem como foco o controle da entrega de tarefas delegadas para os membros.

O primeiro método escolhido foi o *Scrum*, pois o time possui familiaridade. Para o segundo foi escolhido o *Kanban*, um método simples e padrão para o desenvolvimento de *software*.

3.3.1 *Scrum*

O time de desenvolvimento e os orientadores definiram em conjunto a duração das *sprints* de 14 dias. Já as reuniões de planejamento, revisão e retrospectiva foram às quintas-feiras. A equipe também fez *daily meetings*, que são reuniões diárias e curtas.

3.3.2 *Kanban*

O método *Kanban*, detalhado na Seção 2.2.5, necessita da ferramenta *ZenHub*, descrito na Seção 2.5.4, para a sua utilização. Esta ferramenta disponibiliza um quadro de tarefas que permite o controle das funcionalidades que serão desenvolvidas, estão em desenvolvimento, estão sendo revisadas e foram finalizadas. Também permite a visualização de gráficos que demonstram possíveis gargalos no desenvolvimento, facilitando a correção de erros para *sprints* seguintes.

Com isso, o acompanhamento real do desenvolvimento do projeto é facilitado e permite uma comunicação mais clara entre os membros da equipe. O time separou o quadro nas quatro divisões padrões da ferramenta, citadas no parágrafo anterior. Portanto, alterações no quadro não foram necessárias.

3.4 Síntese e apresentação

Na etapa de síntese e apresentação foram gerados todos os artefatos criados nas etapas anteriores, sendo eles o produto de *software* em si e toda sua documentação. Ambos serão tratados nesse documento, além de existirem na plataforma *github*.

A documentação contém toda a parte de arquitetura do produto, *backlog*, gerenciamento e artefatos provenientes de métodos anteriores. A parte do produto de *software* possui todo o código, assim como a configuração do ambiente de desenvolvimento.

Com isso os métodos das etapas de planejamento e identificação de características do sistema foram finalizados e os que serão utilizados na etapa de desenvolvimento tiveram

suas especificações definidas. A partir disso a equipe pode prosseguir para a criação do *backlog*.

4 Requisitos

Os requisitos do *software* são o guia para o desenvolvimento de um produto. Sua definição formal ocorre após o planejamento, e utiliza os artefatos gerados no Capítulo 3 para criação do *backlog*.

Para a construção do *backlog* do produto, foram aplicados dois métodos já apresentados no Capítulo 3, a *Lean Inception* e o PBB. Por fim, os requisitos foram formalizados em formato de história de usuário.

4.1 Definição das características do produto

Para começar a etapa de requisitos foi utilizada a *Lean Inception*, que possui nove artefatos gerados de maneira progressiva. Todos os artefatos existem com o o propósito final da criação do MVP, vide Seção 2.2.1.

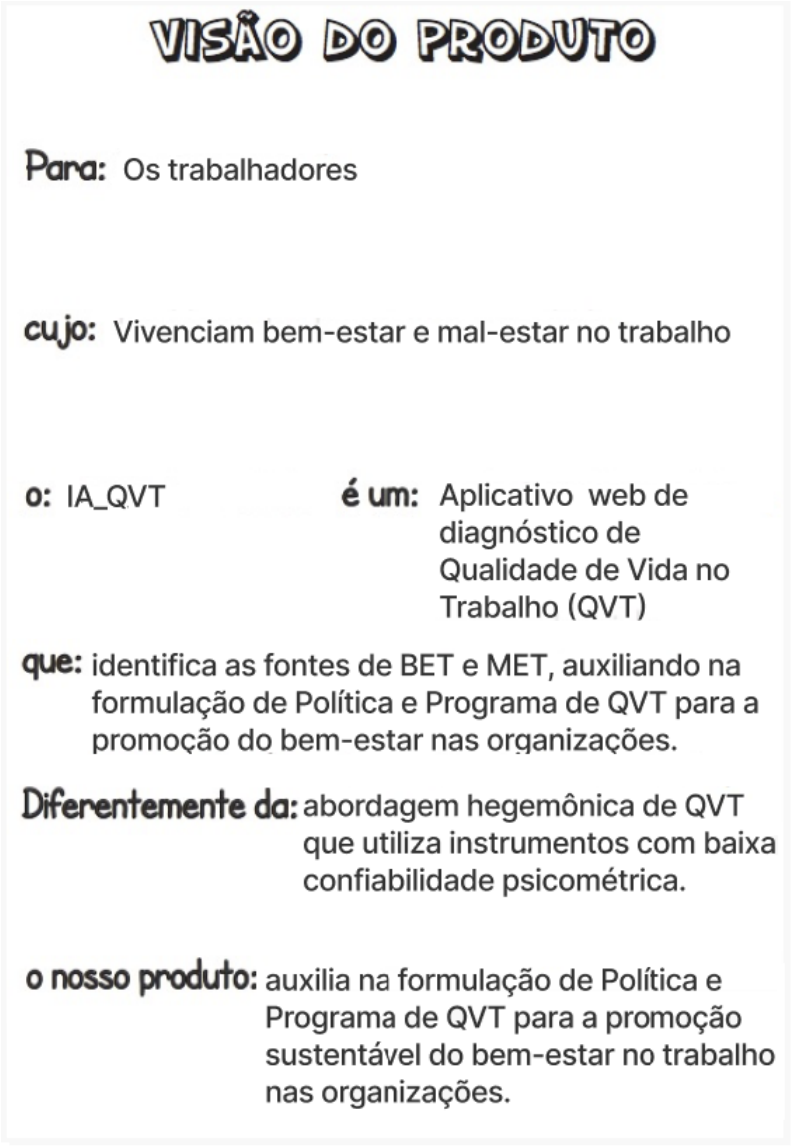
Alguns artefatos são mais relevantes para a identificação de requisitos e criação do *backlog*, pois conversam diretamente com o PBB. Entre eles podem ser encontrados os artefatos que definem a visão e o entendimento do produto, os que definem o usuário e, por fim, os que definem as funcionalidades.

4.1.1 Visão do produto e objetivos

O primeiro artefato a ser mencionado é o artefato de visão do produto, que é desenvolvido para facilitar um entendimento alinhado sobre o produto entre o time e o cliente.

A equipe utilizou o artefato preenchido pelo cliente, vide Figura 5. Após a criação do artefato de visão do produto, o time seguiu para a próxima etapa do método.

Figura 5 – Artefato de visão do produto



Fonte: Os próprios autores (2022a)

Para a construção do artefato de objetivos do produto, cada membro da equipe pontuou algumas características deste, de acordo com o que foi compreendido na atividade de visão do produto.

Ao finalizar o alinhamento do entendimento do produto, é necessário alinhar os objetivos do negócio. Então questiona-se: quais são seus pontos principais? O que faz o produto se destacar?

Para responder ambas as perguntas foi criado o próximo artefato da *Lean Inception*, chamado objetivos do negócio, representado na Figura 6.

Figura 6 – Objetivos de negócio final

Fonte: Os próprios autores 2022a

O artefato apresentado na Figura 6 contém cinco *clusters*. Os *clusters* são os grupos em que as funcionalidades são divididas. Eles também foram utilizados como base para os épicos do *backlog*.

Com os artefatos criados, o entendimento do time sobre o produto foi alinhado. Além do entendimento, o time possuía uma versão mais básica da lista de funcionalidades.

4.1.2 Usuários

Nesse momento, o time possui os artefatos e o entendimento do produto, mas o público-alvo ainda não é conhecido. Para conhecer os usuários e utilizá-los no método, são realizadas duas atividades: as personas e as jornada do usuário.

Após uma breve explicação do cliente sobre quem são os usuários do sistema, foram criados os artefatos do tipo persona.

Figura 7 – Persona do usuário gestor

Fonte: Os próprios autores (2022a)

Na Figura 7 é apresentado o artefato do usuário gestor e as especificações necessárias para sua construção. O artefato persona é descrito em mais detalhes na Seção 2.2.1. Além do gestor, foram criados mais quatro tipos de usuários, totalizando cinco:

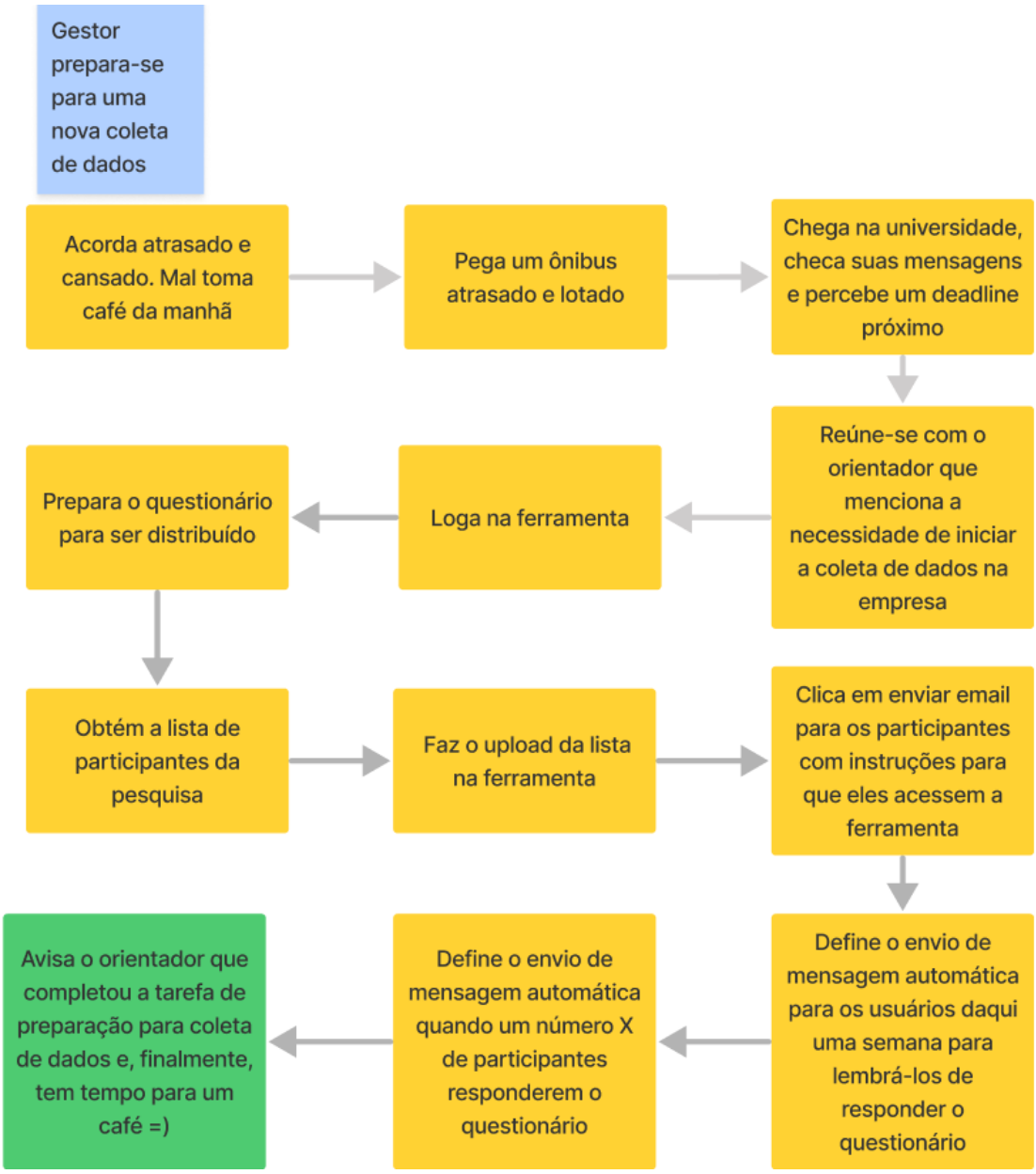
- Servidor: funcionário público concursado, usuário mais comum da aplicação.
- Gestor: pesquisador que vai utilizar o produto para aplicar a metodologia de QVT, a pedido do órgão público ou empresa.
- Terceirizado: trabalhador de empresas que prestam serviços não-primários para o órgão público ou empresa.
- Estagiário: estudante de alguma graduação que trabalha no local.
- Menor Aprendiz: usuário menos comum da aplicação, parecido com o estagiário. Foi escolhido representá-lo como pessoa com deficiência (PcD), pois existem usuários

PcDs no sistema atual e é de extrema importância que o produto seja acessível para todos.

Depois de criar os usuários, é importante entender como cada um vai utilizar e se comportar dentro do sistema. Para isso o time fez o próximo artefato da *Lean Inception*, a jornada de usuários.

A Figura 8 mostra o artefato de jornada de um usuário do tipo gestor. Nele é detalhada a jornada do usuário para o preparo de uma nova coleta de dados. Mais detalhes do artefato são encontrados na Seção 2.2.1.

Figura 8 – Jornada do usuário gestor



Fonte: Os próprios autores (2022a)

A jornada da Figura 12 demonstra o processo do gestor, desde o momento em que acorda até finalizar o preparo para uma coleta de dados na plataforma. Foi criada uma jornada para cada usuário. Delas foram retiradas funcionalidades que comprovam a importância de enxergar o produto com os olhos do seu usuário final.

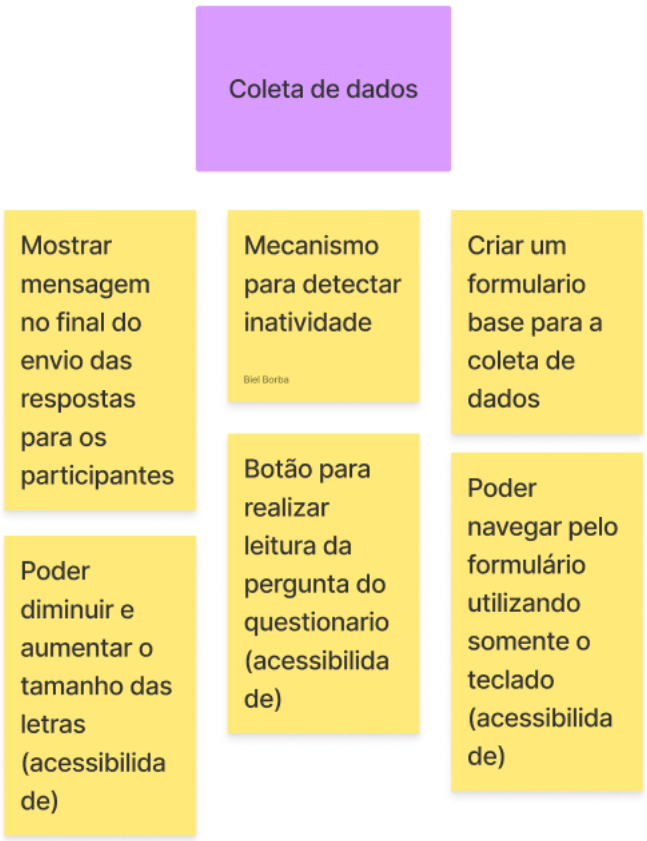
Para entender as funcionalidades derivadas das jornadas, pode ser observado na Figura 8 algumas funcionalidades explícitas que foram utilizadas, posteriormente, na atividade de *brainstorming*, como o envio automático de mensagem com base na porcentagem de respostas.

Com isso, foram criados mais dois artefatos e neste momento o time já possui conhecimento do seu público-alvo, o que possibilita avançar na escrita das funcionalidades.

4.1.3 Funcionalidades

Após alinhar o entendimento, fazer um esboço das funcionalidades e conhecer os tipos de usuário, as funcionalidades foram detalhadas. A equipe escreveu 39 funcionalidades. Foram utilizados os mesmos *clusters* do segundo artefato e um adicional para usabilidade. Os cartões foram separados por tipo: requisitos funcionais em amarelo e não-funcionais em azul.

Figura 9 – Funcionalidades do *cluster* de coleta de dados



Fonte: Os próprios autores (2022a)

Observando a Figura 9, existem funcionalidades que foram criadas se baseando em usuários específicos. Um exemplo disso são as funcionalidades de acessibilidade, voltadas aos usuários PcD.

Verifica-se, então, a importância das atividades de persona e jornada do usuário (vide Seção 4.1.2), pois funcionalidades específicas poderiam passar despercebidas, caso a equipe não conhecesse o usuário do seu produto.

O artefato demonstrado na Figura 9 utilizou dos *clusters* criados durante a atividade de objetivos de negócio (vide Seção 4.1.1) para separar as tarefas. Cada *cluster* é um agrupamento de funcionalidades de um conjunto específico. Nesse artefato, foram criadas as micro-tarefas das quais cada *cluster* será composto.

Após essa atividade, as funcionalidades foram preparadas para o detalhamento das funcionalidades e para o *backlog*. A *Lean Inception* foi realizada na ferramenta *Figma*[1].

4.2 Detalhamento das funcionalidades

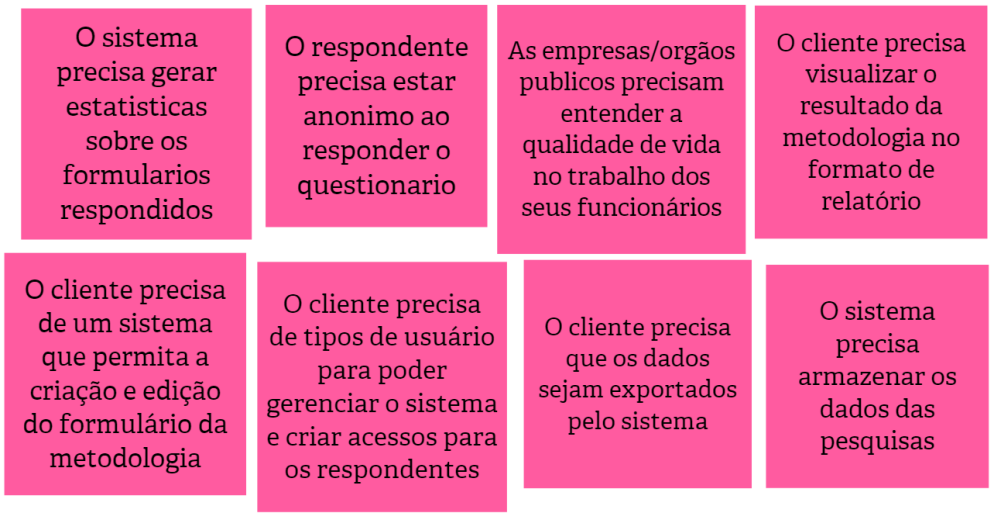
Para o detalhamento das funcionalidades e encaminhamento do *backlog*, foi utilizado o PBB (vide Seção 2.2.2) que possui cinco artefatos. Dos cinco, quatro deles são caminhos para a criação do artefato final, o *product backlog items* (PBI). Todos artefatos desenvolvidos durante o PBB são voltados para o detalhamento dos requisitos, então, todos serão apresentados nessa Seção.

4.2.1 Problemas e Expectativas

No começo do método, a equipe preenche dois artefatos: problemas e expectativas. Ambos funcionam apenas como blocos para a construção dos artefatos seguintes.

[1] Para acessar todos os artefatos gerados na *Lean Inception* acesse: <https://www.figma.com/file/NpoDM4RXmGjvoYXZEVndKB/Lean-Inception-TCC-IA-QVT?node-id=104%3A1251>

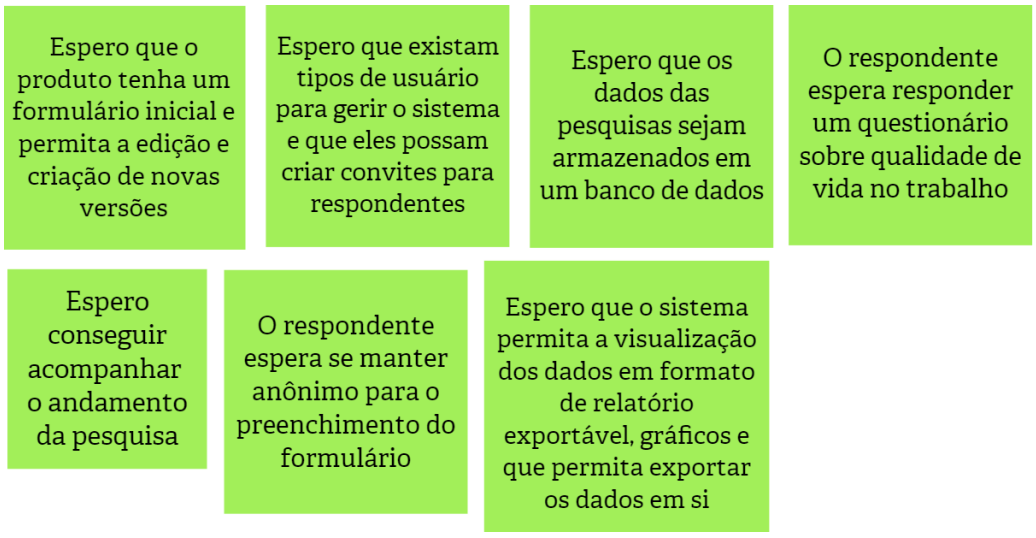
Figura 10 – Problemas, primeiro artefato do PBB



Fonte: Os próprios autores (2022b)

O artefato de problemas, apresentado na Figura 10, contém uma lista de problemas que a equipe de desenvolvimento trouxe para serem solucionados pelo produto de *software*.

Figura 11 – Expectativas



Fonte: Os próprios autores (2022b)

Após apresentar os problemas, foram apresentadas as soluções no artefato de expectativas, evidenciado na Figura 11.

Problemas e expectativas serão utilizados nos artefatos posteriores. Com isso, a equipe possui os blocos de construção das próximas atividades.

4.2.2 Personas

Continuando com o PBB, foi feito o artefato de personas. O artefato de persona é compartilhado com a *Lean Inception*, detalhada na Seção 2.2.1, portanto, a equipe utilizou as personas criadas anteriormente para evitar retrabalho.

As personas, chamadas também de tipos de usuário, são: administrador, gestor e respondente. Todas as personas que apenas respondem o questionário foram agrupadas na categoria de respondente.

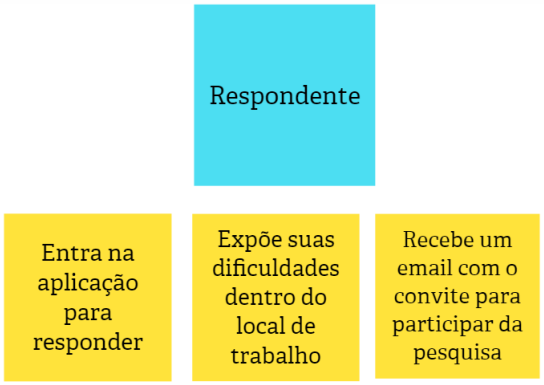
A primeira persona a ser descrita é o gestor. Ele aplica as pesquisas, utiliza dos dados da plataforma e a gerencia, como evidenciado na Figura 12.

Figura 12 – Persona do tipo gestor

Fonte: Os próprios autores (2022b)

Já o respondente apenas acessa a plataforma e responde a aplicação da metodologia, vide Figura 13.

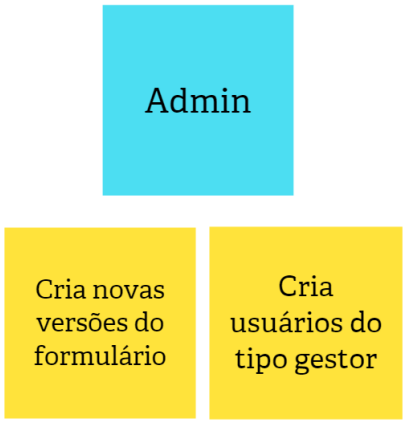
Figura 13 – Persona do tipo respondente



Fonte: Os próprios autores (2022b)

Por fim, o administrador tem como objetivo o controle dos usuários do tipo gestor, além da possibilidade de criar novas versões do formulário, como pode ser visto na Figura 14.

Figura 14 – Persona do tipo administrador



Fonte: Os próprios autores (2022b)

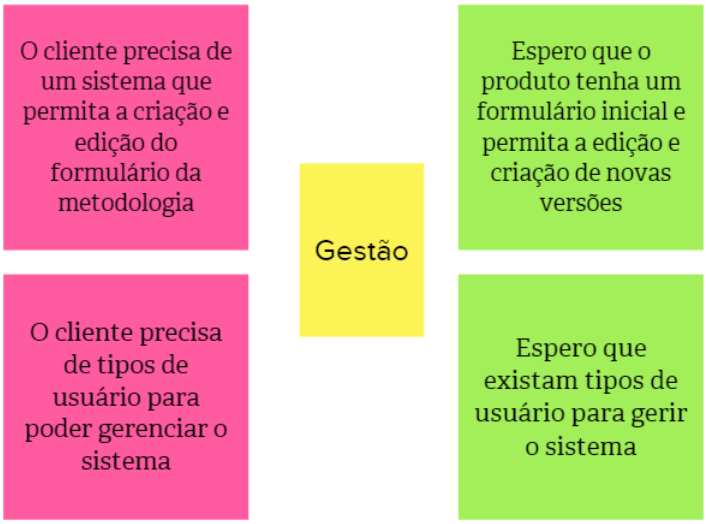
Com isso, todas as personas foram evidenciadas, assim como, suas expectativas e ações que realizarão dentro da plataforma. A partir disso, a equipe pode construir os artefatos finais da metodologia.

4.2.3 *Features* e PBI

Após a definição das personas, foram criadas as *features*, com base nos *clusters* da *Lean Inception*, que podem ser encontrados na Seção 4.1.1. A equipe decidiu reutilizar os *clusters*, pois foram criados em conjunto com o cliente. Por fim, a equipe separou as *features* em micro-tarefas chamadas de *product backlog items* (PBI).

As *features* foram dividas em: gestão, coleta, armazenamento, tratamento e visualização de dados.

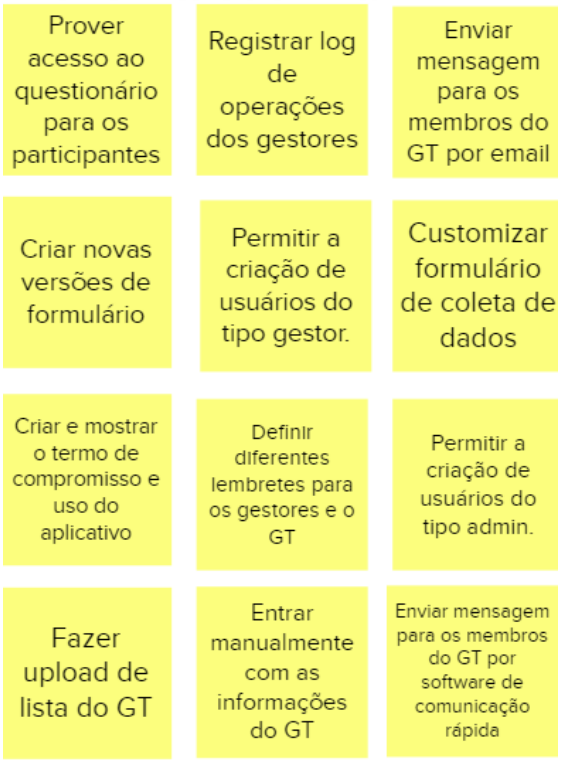
Figura 15 – Macro-tarefa de gestão



Fonte: Os próprios autores (2022b)

Na Figura 15, pode ser visto um exemplo de como ficaram as *features*. Cada *feature* utiliza dos problemas e expectativas criados no começo do método.

Figura 16 – PBIs de gestão



Fonte: Os próprios autores (2022b)

Na Figura 16, está o PBI da *feature* de gestão. São doze micro-tarefas diferentes e cada uma representa uma função específica do sistema. No total, foram criadas 28 micro-tarefas divididas entre cinco *features*. O PBB e seus artefatos foram feitos na ferramenta *Mural*[2].

4.3 *Backlog* do produto

Com os artefatos gerados durante os métodos anteriores foi possível criar o *backlog*. Ele foi escrito no formato de histórias de usuário e épicos, conceitos explicados na Seção 2.2.3. O *backlog* contém todas as histórias definidas para o produto, mesmo as que não entram no escopo do MVP e consequentemente, não desenvolvidas pelo time. No total foram criados quatro épicos, sendo eles:

- **EP01 Gestão**: Gerenciamento da plataforma, desde a aplicação da metodologia até a criação de usuários.

[2] Para acessar todos os artefatos gerados no PBB acesse: <https://app.mural.co/t/tcciaqvt4561/m/tcciaqvt4561/1647013533650/1246ce7f1a006217ce677e2cbeaea3e0c01b6c61?sender=u4279aac9c4b5e91d70730482>

- **EP02 Coleta e armazenamento de dados**: Coleta de dados provenientes da aplicação da metodologia.
- **EP03 Tratamento de dados**: Utilização de técnicas estatísticas para tratar dados, além de realizar sua exportação para outras ferramentas de tratamento.
- **EP04 Visualização de dados**: Criação de gráficos e relatórios para estudos e acompanhamento das aplicações da metodologia.

Além dos épicos, foram criadas 30 histórias de usuário. Para detalhar o épico de gestão, foram criadas treze histórias de usuário. O épico contém todas as histórias relacionadas ao gerenciamento da metodologia, criação de usuários, acesso ao questionário, comunicação com o grupo de trabalho e controle de usuários, como pode ser visto na Tabela 3.

Como explicado na Seção 2.2.3, a pontuação das histórias de usuário foi feita com base em uma história específica, na qual o time de desenvolvimento se reuniu e chegou num consenso. A partir dessa história base, todas as outras foram pontuadas por comparação de três características: complexidade, risco e repetição. A história base definida pelo time foi a US10, *Login* na plataforma.

Tabela 3 – Histórias de usuário do épico de gestão

US	Nome	Descrição	Pontuação
US01	Acesso ao questionário	Eu, como gestor, desejo prover acesso ao questionário para participantes e aplicar a metodologia	3
US02	Entrada manual de informações do GT	Eu, como gestor, desejo entrar manualmente com as informações do GT para que eles possam repassar informações aos respondentes	2
US03	*Upload* de lista com informações do GT	Eu, como gestor, desejo fazer um *upload* com a lista de informações do GT para não precisar escrever manualmente	5
US04	Lembretes	Eu, como gestor, desejo que sejam definidos lembretes para os gestores e para o GT, a fim de controlar a quantidade e o prazo das respostas	8
US05	Envio de *email* para o GT	Eu, como gestor, desejo que sejam enviados *emails* para o GT, para que eles possam repassar informações dos gestores para os respondentes	5

US06	Envio de mensagens para o GT	Eu, como gestor, desejo que sejam enviados mensagens em *softwares* de comunicação rápida para o GT para que eles possam repassar informações dos gestores para os respondentes	8
US07	Customização de formulários	Eu, como gestor, desejo poder customizar os formulários para poder incluir ou retirar perguntas em aplicações específicas da metodologia	13
US08	Criação de usuários do tipo administrador	Eu, como administrador, desejo poder criar novos usuários do tipo administrador para facilitar a administração da plataforma	8
US09	Criação de usuários do tipo gestor	Eu, como administrador, desejo poder criar novos usuários do tipo gestor para poder aplicar a metodologia	8
US10	*Login* na plataforma	Eu, como usuário, desejo poder *logar* na plataforma para utilizar suas funcionalidades	5
US11	Criação de novas versões do formulário	Eu, como administrador, desejo poder criar novas versões do formulário para avançar a aplicação da metodologia	13
US12	Criação termo de compromisso	Eu, como administrador, desejo que exista um termo de compromisso para que respondentes e gestores saibam como se portar durante a utilização da plataforma	1
US13	*Log* de operações dos gestores	Eu, como administrador, desejo que exista um *log* de cada operação dos gestores para poder ter o controle sobre suas ações na plataforma	8

Fonte: Os próprios autores (2022e)

O épico de coleta e armazenamento de dados contém as tarefas da aplicação da metodologia e o armazenamento dos dados gerados. Contém também, todas as histórias de controle durante a aplicação da metodologia, como a verificação de inatividade. As histórias podem ser observadas na Tabela 4.

Tabela 4 – Histórias de usuário do épico de coleta e armazenamento de dados

US	Nome	Descrição	Pontuação
US14	Formulário base para a coleta	Eu, como gestor, desejo que exista um formulário base para que a metodologia possa ser aplicada	8
US15	Gravação de tempo de resposta	Eu, como gestor, desejo que o tempo de resposta de cada respondente seja gravado para análise posterior	3
US16	Mecanismo para verificação de inatividade	Eu, como gestor, desejo que exista um mecanismo de detecção de inatividade para que o usuário seja desconectado após um certo tempo sem responder as questões do formulário	5
US17	Mensagem de confirmação de envio	Eu, como respondente, desejo receber uma mensagem de confirmação após o término do formulário para ter certeza de que as respostas foram entregues	1
US18	Alteração de tamanho da fonte	Eu, como respondente, desejo poder alterar o tamanho da fonte para conseguir realizar a leitura de certas perguntas, se necessário	2
US19	Navegação do formulário pelo teclado	Eu, como respondente, desejo poder navegar pelo formulário utilizando apenas o teclado para, se necessário	1
US20	Botão de leitura de pergunta	Eu, como respondente, desejo que exista um botão para a leitura de cada questão do formulário, caso haja algum respondente que possua dificuldades ou impedimentos para realizar a leitura	3

Fonte: Os próprios autores (2022e)

O épico de tratamento de dados contém todas as histórias que envolvem aplicações de técnicas estatísticas nos dados. As técnicas podem ser aplicadas dentro ou fora da plataforma, pois o sistema permitirá a exportação dos dados. Todas as histórias do épico podem ser visualizadas na Tabela 5.

Tabela 5 – Histórias de usuário do épico de tratamento de dados

US	Nome	Descrição	Pontuação

US21	Aplicação de técnicas estatísticas para cada tipo de pergunta	Eu, como gestor, desejo que sejam aplicadas técnicas estatísticas com base em cada tipo de pergunta para ajudar durante a análise dos dados	5
US22	Aplicação de outras técnicas estatísticas	Eu, como gestor, desejo poder aplicar novas técnicas estatísticas em certas questões, se necessário, para ajudar na análise dos dados	8
US23	Aplicação técnicas estatísticas para detectar respostas inadequadas	Eu, como gestor, desejo que sejam aplicadas técnicas estatísticas para detectar respostas inadequadas	3
US24	Exportação de dados	Eu, como gestor, desejo poder exportar os dados nos formatos *excel*, *csv* e *txt* para poder analisar as respostas em plataformas diferentes	3

Fonte: Os próprios autores (2022e)

A Tabela 6 contém todas as histórias do épico de visualização de dados. O épico agrupa as histórias que cuidam da parte de visualização do resultado da aplicação da metodologia. Os dados serão visualizados por meio de relatórios intermediários em uma página específica contendo alguns gráficos. No final da aplicação, será gerado um relatório final para apresentação.

Tabela 6 – Histórias de usuário do épico de visualização de dados

US	**Nome**	**Descrição**	**Pontuação**
US25	Geração de gráficos base	Eu, como gestor, desejo que sejam gerados gráficos base para cada pergunta, pois estes são necessários para a visualização das respostas	5
US26	Geração de gráficos adicionais	Eu, como gestor, desejo poder gerar gráficos adicionais para as perguntas pois podem complementar a análise	8
US27	Filtrar dados	Eu, como gestor, desejo poder pesquisar por diferentes aplicações, utilizando filtros para conseguir visualizar os dados coletados	5

US28	Geração de relatórios intermediários	Eu, como gestor, desejo que o sistema gere relatórios intermediários para facilitar o acompanhamento da aplicação da metodologia	8
US29	Geração de relatório final	Eu, como gestor, desejo que o sistema gere um relatório final em algum formato de *slide* para enviá-lo aos supervisores da pesquisa	13
US30	Visualização de respostas	Eu, como gestor, desejo visualizar o resultado final das aplicações da metodologia para analisar quantitativa e qualitativamente as respostas	8

Fonte: Os próprios autores (2022e)

Com isso, todos os épicos possuem suas histórias de usuário definidas. O épico com mais histórias é o de gestão, possuindo treze. O épico de coleta e armazenamento possui sete histórias. Já o épico de tratamento possui quatro histórias de usuários. Por fim, o épico de visualização possui seis. Os critérios de aceitação se encontram no repositório do projeto que, a pedido do cliente, é privado.

Após à definição das histórias o time pode começar o desenvolvimento. No próximo capítulo será abordado o processo de desenvolvimento do sistema e suas especificidades.

5 Desenvolvimento do sistema

Este capítulo se refere a execução da proposta apresentada no Capítulo 1, correspondendo aos objetivos indicados na Seção 1.1 deste trabalho. Aqui, será abordado o processo de desenvolvimento do sistema proposto, assim como as dificuldades e alterações necessárias realizadas sobre a proposta inicial.

Para iniciar o desenvolvimento do sistema é necessário definir os métodos de apoio que serão utilizados e qual arquitetura utilizada para o desenvolvimento. Primeiro, foram definidos e justificados os nossos métodos de apoio na Seção 3 e, nessa seção, será discutido o planejamento dos riscos e da arquitetura do sistema.

5.1 Planejamento de Riscos

Essa subseção tem como objetivo identificar e explicar os riscos acerca do desenvolvimento do produto e também definir seus níveis de preocupação.

5.1.1 Identificação dos Riscos

Para identificação dos riscos o grupo utilizou os encontros realizados de dois em dois dias e as reuniões de revisão e retrospectiva. Foi realizada a criação de um formulário para ser passado na reunião de retrospectiva com o intuito de permitir os membros demonstrarem suas percepções obtidas durante a *sprint*.

A quantificação do risco foi feita por meio do uso da seguinte fórmula: **impacto * probabilidade = nível de preocupação**. À partir dessa identificação pode-se perceber quais são os riscos que demandam mais atenção. Os riscos são classificados em quatro categorias sendo elas: técnico, organizacional, pessoal e de requisitos. Eles são pontuados com relação a seu impacto e probabilidade, na Tabela 7 podemos observar os impactos e na Tabela 8 as probabilidades utilizadas para o cálculo do **nível de preocupação**.

Tabela 7 – Classificação dos impactos

Classificação dos impactos		
Impacto	**Descrição**	**Valor**
Nenhum	Nenhum impacto	0
Muito Baixo	Impacto não significativo	1
Baixo	Impacto de baixa influência	2
Moderado	Impacto notável com poucas consequências	3
Alto	Impacto que compromete o andamento do projeto	4
Muito Alto	Impacto que inviabiliza o andamento do projeto	5

Fonte: Rocha e Neri (2016)

Tabela 8 – Classificação das probabilidades

Classificação das probabilidades		
Probabilidade	**Descrição**	**Valor**
Nenhuma	Nenhuma	0
Muito Baixa	Menos de 20%	1
Baixa	21% à 40%	2
Moderada	41% à 60%	3
Alta	61% à 80%	4
Muito Alta	Mais de 80%	5

Fonte: Rocha e Neri (2016)

A Figura 17 mostra os riscos levantados pela equipe e suas devidas categorias sendo a maior quantidade de riscos pessoais, seguidos por riscos técnicos e por fim, riscos organizacionais.

Figura 17 – Riscos levantados

Quando foi Identificado	Descrição	Categoria
Sprint 1	Sobrecarga de algum membro	Pessoal
Sprint 1	Desistência	Pessoal
Sprint 1	Negligência	Pessoal
Sprint 1	Problemas Pessoais	Pessoal
Sprint 1	Falta de conhecimento em alguma tecnologia	Técnico
Sprint 1	Surgimento de novos requisitos	Organização
Sprint 1	Erros e problemas no desenvolvimento	Organização
Sprint 1	Retrabalho por entregas mal feitas	Técnico
Sprint 2	Baixa granularidade das tarefas	Técnico

Fonte: Os próprios autores (2022a)

Em geral os riscos foram rapidamente identificados sendo oito deles identificados logo na primeira *sprint* e apenas um identificado na segunda *sprint*, que foi a *sprint* onde se iniciou o desenvolvimento do produto.

A classificação dos impactos e probabilidades dos riscos levantados pela equipe está disponível no *Google Drive*[1].

5.2 Planejamento da Arquitetura

Um monólito é uma aplicação de *software* em que os módulos não podem ser executados independentemente. A seguir são apresentados os motivos, de acordo com Dragoni et al. (2017), do porquê não foi utilizada a arquitetura monolítica.

Primeiramente, os monólitos de grande porte são difíceis de manter e evoluir devido à sua complexidade, além disso, rastrear *bugs* requer longas leituras através de sua base de código. Também existe a questão de que qualquer alteração em um módulo de um monólito requer a reinicialização de toda a aplicação. Para projetos de grande porte, a reinicialização geralmente acarreta em tempos de inatividade consideráveis, dificultando o desenvolvimento, teste e manutenção do projeto.

Os monólitos também limitam a escalabilidade. A estratégia usual para lidar com incrementos de requisições de entrada é criar novas instâncias da mesma aplicação e

[1] <https://docs.google.com/spreadsheets/d/1DBacSUPWWWaPnyExdczFPyL1W8v0VruohVI8dsZPM08/edit?usp=sharing>

dividir a carga entre elas. No entanto, pode ser que o aumento do tráfego esforce apenas um subconjunto dos módulos, tornando inconveniente a alocação dos novos recursos para os outros componentes. Além de representarem um aprisionamento tecnológico para os desenvolvedores, que são obrigados a usar a mesma linguagem e estruturas da aplicação original.

O estilo arquitetural de microsserviços foi proposto para lidar com essas limitações do estilo monolítico e para entendê-lo melhor, é essencial conhecer as seguintes definições: microsserviços e arquitetura microsserviço. A primeira, é definida como um processo coeso e independente que interage por meio de mensagens. Já a segunda é uma aplicação distribuída onde todos os seus módulos são microsserviços (DRAGONI et al., 2017). A seguir são citadas algumas das vantagens de se utilizar a arquitetura microsserviço, de acordo com Dragoni et al. (2017), que é a arquitetura escolhida para o desenvolvimento do produto.

Os microsserviços implementam uma quantidade limitada de funcionalidades, o que torna sua base de código pequena e limita, inerentemente, o escopo de um *bug*. Além disso, como os microsserviços são independentes, um desenvolvedor pode testar e investigar diretamente suas funcionalidades de forma isolada em relação ao restante do sistema. Além disso, é possível planejar transições graduais para novas versões de um microsserviço. A nova versão pode ser implantada "ao lado" da antiga e os serviços que dependem desta última podem ser gradualmente modificados para interagir com a primeira. Isso promove a integração contínua e facilita a manutenção do *software*.

Como consequência do item anterior, a alteração de um módulo de uma arquitetura de microsserviço não requer uma reinicialização completa de todo o sistema. A reinicialização diz respeito apenas aos microsserviços deste módulo. Por serem pequenos em tamanho, os programadores podem desenvolver, testar e manter serviços com tempos de inatividade de reimplantação muito curtos. Além disso, a única restrição imposta em uma rede de microsserviços interoperantes é a tecnologia usada para fazê-los se comunicar (mídia, protocolos, codificações de dados). Por fim, os microsserviços não impõem bloqueios adicionais e os desenvolvedores podem escolher livremente os recursos ideais (linguagens, *frameworks*, etc.) para a implementação de cada microsserviço (DRAGONI et al., 2017).

A partir dessas informações e de discussões entre os participantes do desenvolvimento do software foi desenvolvido o documento de arquitetura presente no Apêndice A.

5.3 Execução das *Sprints*

De acordo com Coutinho (2018), *sprints* são ciclos de trabalho, utilizados no método *Scrum* (detalhado na Seção 2.2.4), em que algum valor é acrescentado ao produto desenvolvido. O método é dividido em uma quantidade de *sprints* definidas pelo time, que seguem de maneira consecutiva. No caso desse produto, as *sprints* foram de 14 dias.

5.3.1 Primeira *Sprint*

A primeira *sprint* realizada pelo time contou com sete tarefas, sendo todas de preparação para o desenvolvimento do código e documentação. As tarefas planejadas foram divididas em duas categorias, documentação e preparação de ambiente e seus resultados se encontram na Tabela 9.

Tabela 9 – *Sprint Backlog*

Sprint Backlog		
Issue	**Pontuação**	**Concluído**
Criar o documento de arquitetura	0	Sim
Preparar o local para armazenar a documentação	0	Sim
Subir o *backlog* no repositório	0	Sim
Subir artefatos metodológ no repositório	0	Sim
Preparar o ambiente do repositório de questionário	0	Sim
Preparar o ambiente do repositório de perfil	0	Sim
Preparar o ambiente do repositório de *frontend*	0	Sim

Fonte: Os próprios autores (2022f)

As tarefas de documentação foram: criar o documento de arquitetura, preparar o local para armazenar a documentação, subir o *backlog* no repositório e subir artefatos da *Lean Inception* e do PBB no repositório. De todas as tarefas citadas, a mais importante foi a de criação do documento de arquitetura, pois ele guia o time na modelagem do banco de dados e na criação dos microsserviços. O documento de arquitetura pode ser encontrado no Apêndice A.

Já as tarefas de preparação de ambiente foram: preparar o ambiente do repositório de questionário, de perfil e de *frontend*. Todos os ambientes possuem *Docker*, vide Seção 2.5.3, para virtualizar o ambiente e facilitar o desenvolvimento em diferentes máquinas. Além de possuírem um conjunto de ações do *github* para garantir a qualidade do código. As tarefas foram realizadas sem dificuldade e a primeira *sprint* foi finalizada pelo time, permitindo a conclusão da primeira etapa do trabalho de conclusão de curso.

5.3.2 Segunda *Sprint*

A segunda *sprint* realizada pelo time contou com sete tarefas de código. As tarefas de código foram divididas em dois grupos principais, *frontend* e *backend*, sendo cinco de *frontend* e duas de *backend*.As tarefas planejadas e seus resultados se encontram na Tabela 10.

Tabela 10 – *Sprint 2 Backlog*

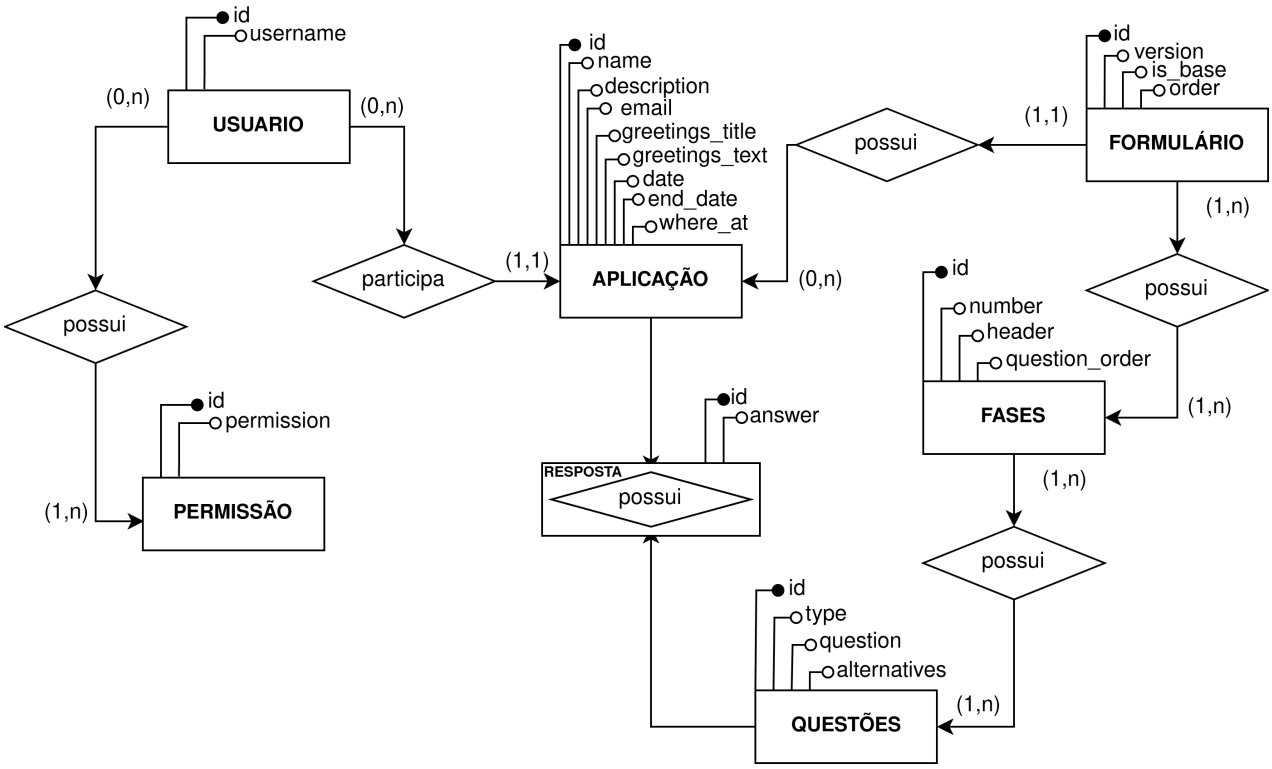
Sprint 2 Backlog		
Issue	**Pontuação**	**Concluído**
Criar o formulário base de coleta *frontend*	3	Não
Criar o formulário base de coleta *backend*	3	Sim
Ajustar tamanho da fonte do questionário	2	Sim
Botão de leitura das questões do questionário	3	Não
Criar e mostrar termo de compromisso de uso	1	Não
Customizar formulário de coleta *frontend*	13	Não
Customizar formulário de coleta *backend*	13	Sim

Fonte: Os próprios autores (2022f)

Para a realização correta das tarefas de *backend* foi necessário refatorar o documento de arquitetura, dado que diagramas continham erros na modelagem. Por conta disso, as alterações nos diagramas foram realizadas durante o desenvolvimento do código. Com isso as histórias de usuário relacionadas ao *backend* foram finalizadas.

As alterações principais se deram em criar uma nova entidade chamada fase, que funciona como um meio termo entre formulário e questão, para separar as questões apresentadas em cada página do formulário. A outra alteração é a criação de uma entidade chamada respostas de aplicação, que possui um relacionamento com questão e aplicação, guardado as respostas de cada questão naquela aplicação específica.

Figura 18 – Diagrama Entidade Relacionamento



Fonte: Rossi e Rocha (2022d)

Fonte: Os próprios autores (2022f)

Para corresponder as mudanças no código, foram alterados os diagramas do documento de arquitetura. Na Figura 18, pode ser visto o diagrama entidade relacionamento com as alterações no banco de dados.

As tarefas designadas ao *frontend* na segunda *sprint* se mostraram mais complexas do que o time havia previsto e por isso, foram classificadas como tarefas em atraso. Com isso a segunda *sprint* foi finalizada e o time pode seguir com o desenvolvimento.

5.3.3 Terceira *Sprint*

A terceira *sprint* continha menos tarefas em comparação com a *sprint* anterior. Todas as tarefas estavam relacionadas ao desenvolvimento de histórias de usuário, sem nenhuma de documentação e preparação de ambiente. As tarefas planejadas e seus resultados se encontram na Tabela 11.

Tabela 11 – *Sprint 3 Backlog*

Sprint 3 Backlog		
Issue	**Pontuação**	**Pontuação**
Criar o formulário base de coleta *frontend*	3	Sim
Botão de leitura das questões do questionário	3	Sim
Criar e mostrar termo de compromisso de uso	1	Sim
Customizar formulário de coleta *frontend*	13	Sim
Exportar respostas para IRaMuTeQ e CSV	3	Sim
Poder navegar pelo formulário usando o teclado	1	Não
Criar tela de estatísticas de pesquisas	5	Não
Aplicar automaticamente técnicas estatísticas *backend*	5	Não
Gerar automaticamente gráficos sobre as respostas	5	Não

Fonte: Os próprios autores (2022f)

Para o *backend*, foram delegadas duas histórias, uma de exportação dos dados da aplicação e outra para a aplicação de técnicas estatísticas. Além delas, durante a *sprint* foi necessário realizar algumas refatorações, que acabou se tornando o trabalho principal. A tarefa de refatoração e exportação foram finalizadas, ficando com apenas a de aplicação estatística como atraso.

Para o *frontend*, foram finalizadas as histórias de criação do formulário, customização do formulário base, botão de leitura das questões do questionário e criação dos termos de compromisso da *sprint* anterior que estavam como atraso. Devido ao atraso da *sprint* anterior, as novas histórias de usuário desta *sprint* acabaram atrasadas. Para mitigar os riscos, a equipe decidiu que a próxima *sprint* seria para resolver as atrasos e pendências, evitando assim o acúmulo de tarefas.

5.3.4 Quarta *Sprint*

A quarta *sprint* foi utilizada para concluir as histórias de usuário que ainda estavam pendentes, principalmente no *frontend*. No *backend*, além de finalizar pendências, foram iniciadas as histórias de criação de usuários do tipo gestores e administradores e acesso ao questionário por parte dos respondentes. Na tabela 12 podem ser vistas as atividades planejadas para a *sprint*.

Tabela 12 – *Sprint 4 Backlog*

Sprint 4 Backlog		
Issue	**Pontuação**	**Concluído**
Poder navegar pelo formulário usando o teclado	1	Sim
Criar tela de estatísticas de pesquisas	5	Sim
Aplicar automaticamente técnicas estatísticas	5	Não
Gerar automaticamente gráficos sobre as respostas	5	Não
Prover acesso ao questionário para os participantes	3	Não
Permitir a criação de usuário administrador	8	Sim
Permitir a criação de usuário gestor	8	Sim

Fonte: Os próprios autores (2022f)

A carga de trabalho para o *backend* estava alta, portanto algumas histórias de usuário foram priorizadas. Como apareceu a necessidade de utilizar os usuários para testes, as histórias relacionadas a criação de usuário foram escolhidas para serem priorizadas, sendo ambas finalizadas nesta *sprint*. A história pendente foi avançada, mas não finalizada, continuando como atraso para a próxima iteração.

No *frontend*, como previsto na *sprint* anterior, foram terminadas as histórias atrasadas de navegação pelo teclado e criação da tela de estatísticas e também iniciada a nova história de criação de usuários. Com isso, a iteração foi finalizada e a próxima foi preparada.

5.3.5 Quinta e Sexta *Sprints*

A equipe realizou mais duas *sprints*, sendo ambas com o objetivo de fechar pendências antigas, histórias de usuário que estavam em aberto como pode ser visto na tabela 13.

Tabela 13 – *Sprint 5 e 6 Backlog*

Sprint 5 e 6 Backlog		
Issue	**Pontuação**	**Concluído**
Aplicar automaticamente técnicas estatísticas	5	Sim
Gerar automaticamente gráficos sobre as respostas	5	Sim
Prover acesso ao questionário para os participantes	3	Sim
Realizar login na plataforma	5	Sim

Fonte: Os próprios autores (2022f)

O trabalho do *backend* se concentrou em duas histórias de usuário, sendo elas a finalização da aplicação das técnicas estatísticas e geração de gráficos e o acesso ao questionário por parte dos respondentes. Também foram realizadas correções pontuais durante as iterações.

Para o *frontend* na quinta *sprint* foi realizada a história de *login* e finalizada a história de criação de usuários e na sexta *sprint* a equipe finalizou a história de prover acesso ao formulário e realizou a padronização da interface gráfica da aplicação.

A sexta *sprint* foi a última realizada pelo time, finalizando assim o desenvolvimento do produto mínimo viável (MVP). O time conseguiu finalizar as histórias de usuário principais para o MVP e entregá-lo dentro do prazo esperado. O próximo capítulo detalha um pouco mais a aplicação desenvolvida e suas funcionalidades.

6 O produto

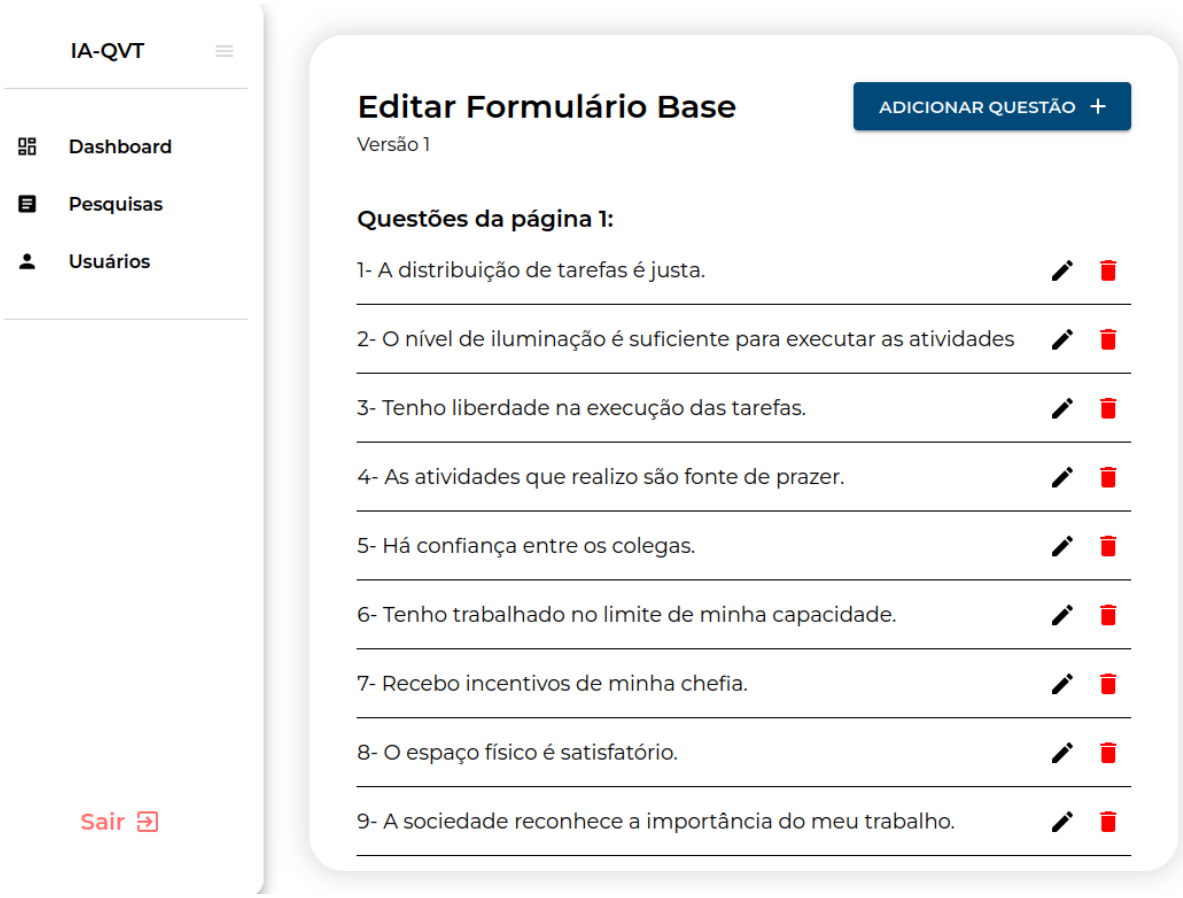
Neste capítulo é detalhado produto final desse trabalho. O capítulo está dividido em quatro subseções, para guiar o leitor sobre o uso do produto, sendo elas: formulários, criação e acompanhamento de pesquisas, questionário acessado pelo respondente e usuários.

6.1 Formulários

O formulário é um *template* utilizado para gerar pesquisas, que contém um conjunto específico de questões e páginas. Por ser um *template*, pode ser reutilizado em diferentes pesquisas sem a necessidade de gerar novamente o conjunto das questões.

Para aplicar a metodologia, é necessário criar uma pesquisa. Para criar uma pesquisa, é necessário possuir um formulário contendo todas as questões necessárias para a mesma. Pensando nisso, a equipe de desenvolvimento criou um *seed* para o formulário base, e a partir dele é possível adicionar, retirar e editar questões, como pode ser observado na Figura 19.

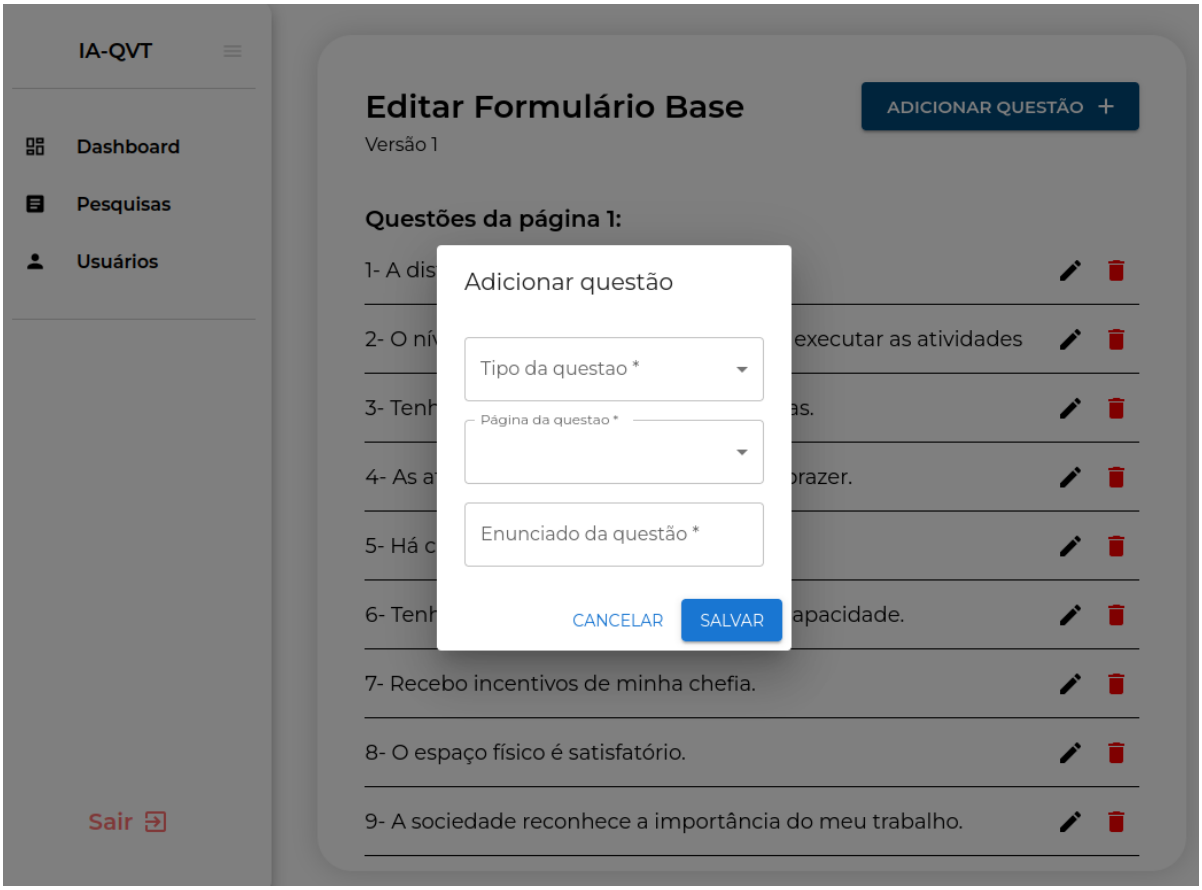
Figura 19 – Tela de edição do formulário base



Fonte: Os próprios autores (2022a)

Como apresentado no parágrafo anterior, a aplicação permite editar, retirar e adicionar questões. Para padronizar o trabalho, as questões foram divididas em diferentes tipos, sendo eles: questões abertas, questões curtas de texto, questões de concordância, questões numéricas, questões de múltipla escolha e questões do tipo *likert*. A partir disso, questões novas podem ser criadas, necessitando apenas de um tipo, a sua página e o enunciado, como pode ser observado na Figura 20.

Figura 20 – Modal de criação de questões



Fonte: Os próprios autores (2022a)

Com a implementação dessa página, os usuários podem realizar quaisquer ações necessárias para a criação de uma pesquisa, seja editando o formulário base ou utilizando-o da maneira que se encontra. Por fim, após preparar o formulário, o usuário pode progredir à próxima ação do fluxo: a preparação e criação de uma pesquisa.

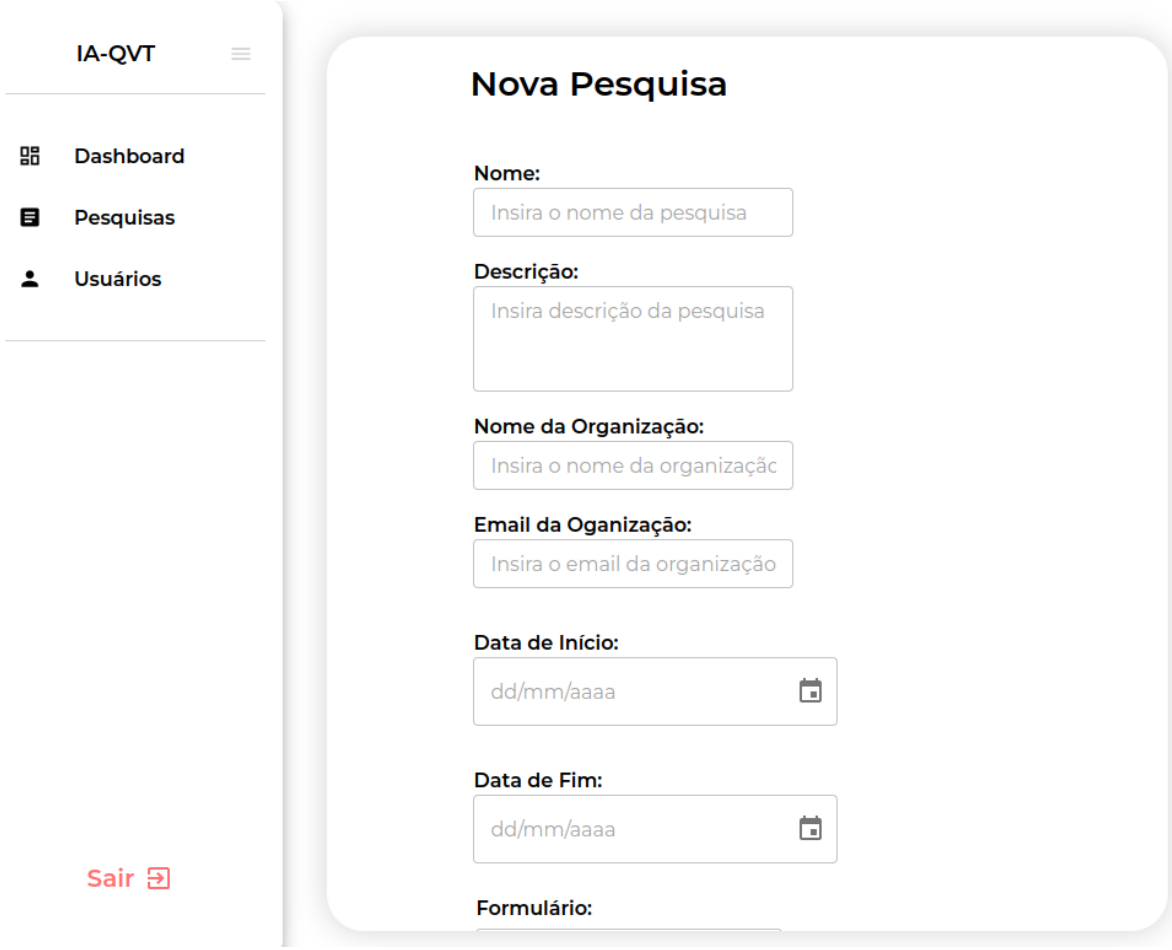
6.2 Criação e acompanhamento de uma pesquisa

A pesquisa é a aplicação de um formulário para um cliente específico. O mesmo formulário aplicado em locais diferentes constitui diferentes pesquisas, portanto podem existir múltiplas pesquisas por *template* de formulário. Com isso, é aberta a possibilidade de pesquisas acontecerem simultaneamente.

Para gerar cada pesquisa, é necessário um conjunto de atributos que a identificam, sendo eles: nome, descrição, nome da organização, *e-mail* da organização, data de início e fim da pesquisa, formulário que será utilizado, título que irá na tela de agradecimento, texto da tela de agradecimento e o conjunto de informações do grupo de trabalho da organização. O grupo de trabalho é um conjunto de pessoas internas que ficará responsável

por ajudar a equipe que realizará a aplicação da pesquisa. Todos esses campos podem ser visualizados na Figura 21.

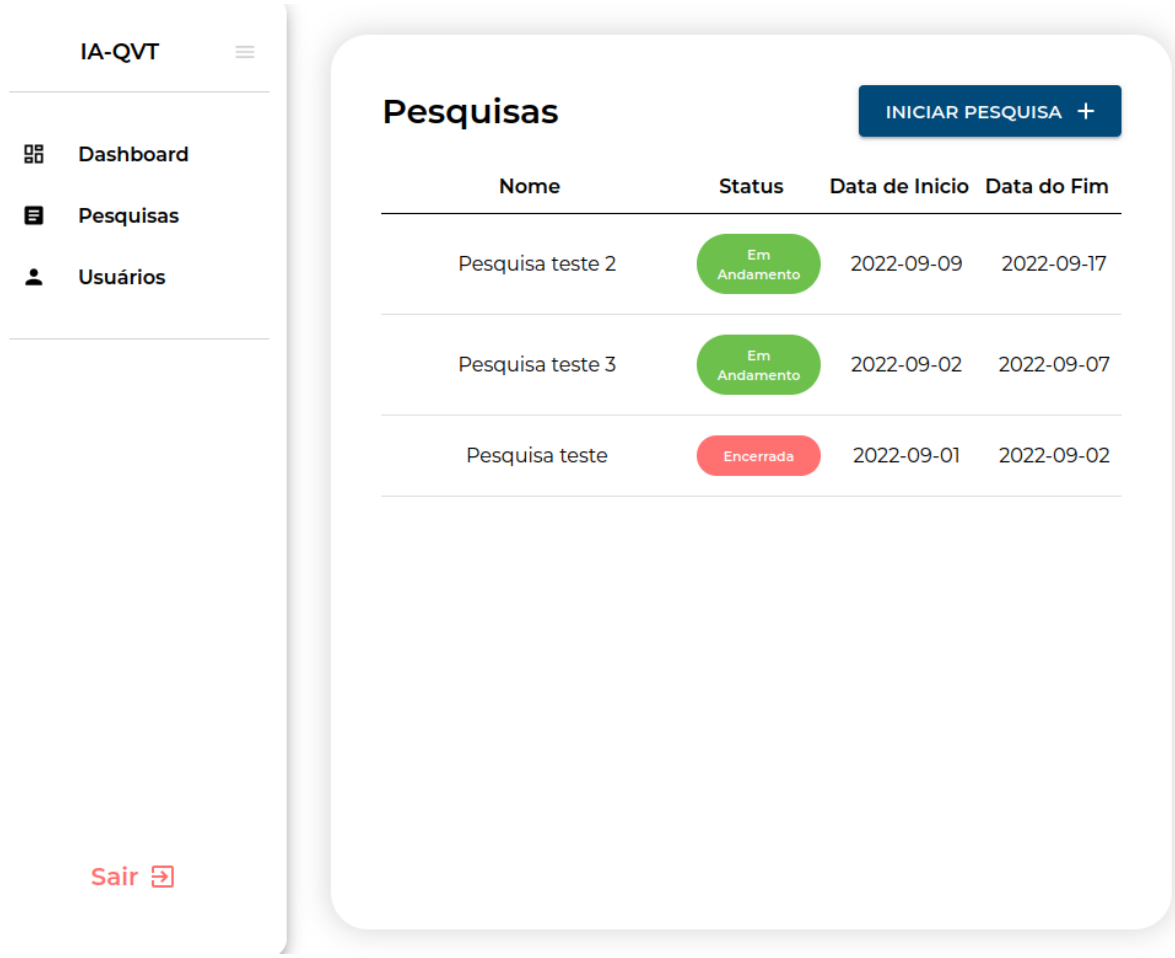
Figura 21 – Tela de criação de pesquisas



Fonte: Os próprios autores (2022a)

Após criar a pesquisa, os gestores e administradores que compõe a equipe de aplicação precisam acompanhar os resultados da mesma. Para isso foram criadas duas telas de acompanhamento. A primeira é composta por uma lista contendo pesquisas em aberto, seguido das finalizadas, como pode ser observado na Figura 22. Essa tela permite que os gestores e administradores visualizem quaisquer pesquisas da plataforma e o seu *status*.

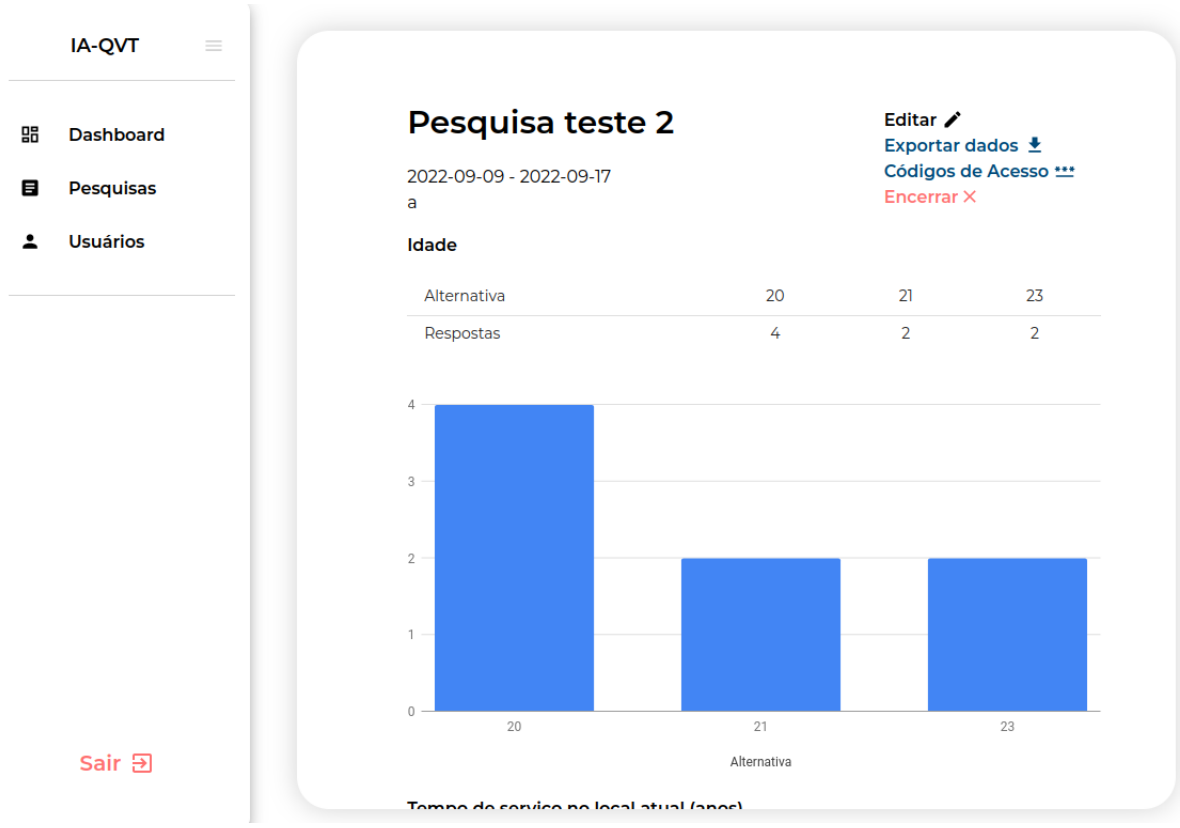
Figura 22 – Tela de pesquisas gerais



Fonte: Os próprios autores (2022a)

A partir da tela geral de pesquisas, o usuário decide acompanhar os resultados de uma pesquisa em aberto ou visualizar resultados de pesquisas finalizadas. Para isso, foi criada a tela específica de uma pesquisa. Nesta tela é possível acompanhar como está a distribuição de respostas em questões com alternativas. Além disso, a plataforma permite a exportação dos dados nos formatos de *csv*, *xlsx* e *txt*. A tela pode ser vista na Figura 23.

Figura 23 – Tela de resultados de uma pesquisa



Fonte: Os próprios autores (2022a)

Os arquivos nos formatos *csv* e *xlsx* contém apenas questões com alternativas, como múltipla escolha e *likert*. Já os arquivos em formato *txt* contém os dados de questões abertas para serem analisadas por *softwares* específicos de análise de conteúdo, discurso e lexicometria. Para facilitar essa análise, ao selecionar a exportação por *txt*, cada questão aberta resultará em um arquivo. Portanto, o usuário recebe um arquivo *zip* contendo o arquivo *txt* específico de cada questão.

Para coletar os resultados, a pesquisa deve ser respondida pelos funcionários da organização que a mesma for aplicada. Para isso foi criada a página de geração de código, na qual são criados identificadores únicos para uma quantidade específica de respondentes. Ao gerar o identificador, a equipe de aplicação deve selecionar a quantidade de códigos necessários e enviar o arquivo *csv* gerado ao GT, que os distribuirá.

6.3 Usuários

Na aplicação existem três tipos de usuários, o administrador, o gestor e o respondente. A interação com o usuário gestor ou administrador se da inicialmente pela página de *login* onde são inseridas as informações necessárias para sua autenticação como pode

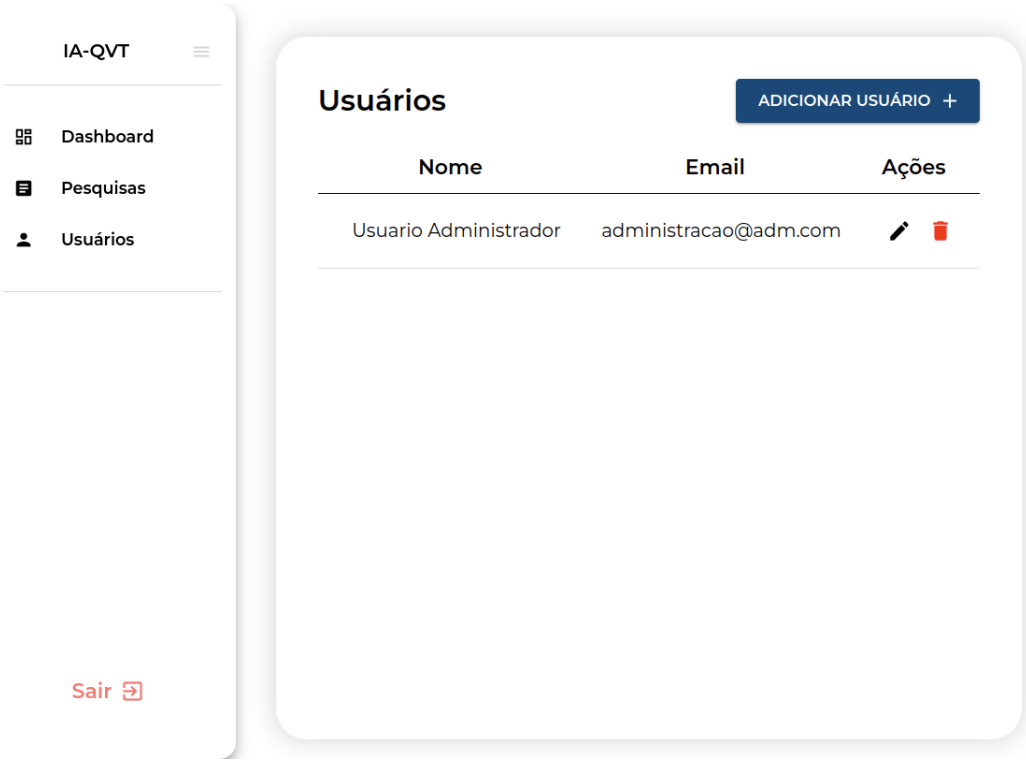
ser visto na Figura 24.

Figura 24 – Tela de Login

Fonte: Os próprios autores (2022a)

As informações necessárias para autenticação são *email* e senha, sendo que o *email* é guardado como chave única no banco de dados para evitar incongruências nas informações. Na tela de *login* não há botão para registrar, pois ao iniciar o banco de dados já é realizada a criação de um usuário administrador padrão e a partir dele que é possível registrar novos usuários na aplicação, como pode ser visto na Figura 25.

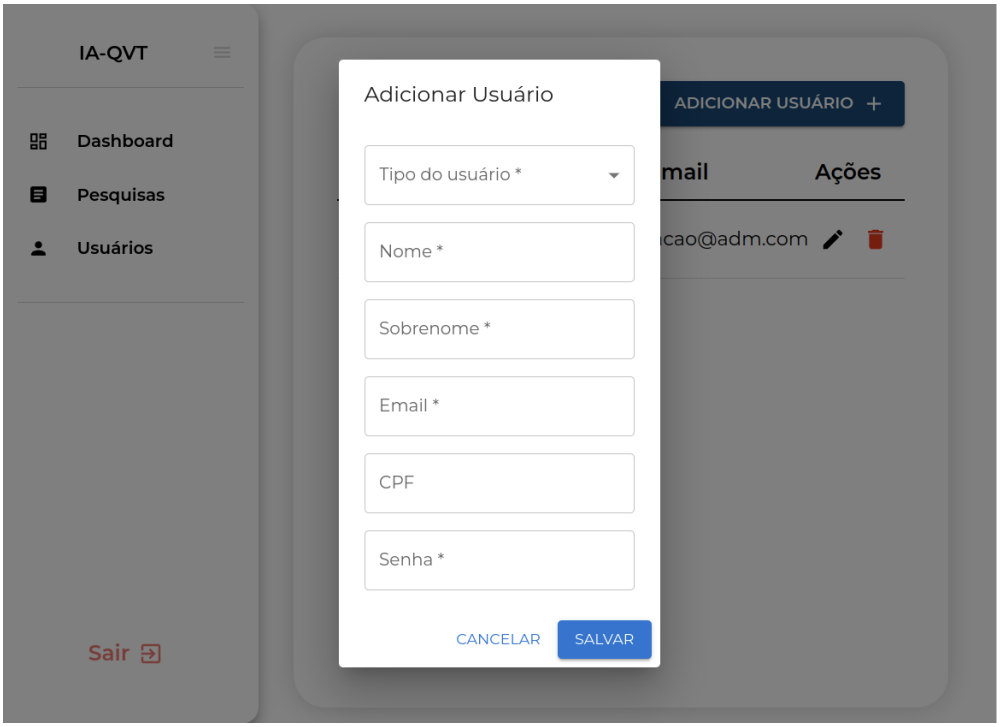
Figura 25 – Tela de Usuários



Fonte: Os próprios autores (2022a)

É na tela de usuários que o administrador do sistema pode cadastrar, editar e remover usuários. O cadastro de usuários se da à partir da modal representada na Figura 26 onde é necessário preencher o tipo do usuário que está sendo criado, gestor ou administrador, o nome, sobrenome, cpf, *email* e senha.

Figura 26 – Modal de criação de usuários



Fonte: Os próprios autores (2022a)

A edição também é feita na mesma modal da Figura 26, porém ao editar, os campos são mostrados com os dados do usuário já cadastrado com exceção do campo de senha por questões de segurança.

Por fim, a remoção de usuários já cadastrados é feita por meio de um diálogo de confirmação sobre a exclusão, que garante a ciência da ação permanente do administrador.

A interação com o usuário respondente se dá inicialmente por meio da tela de introdução e posteriormente pelas telas do questionário. A Figura 27 mostra a página de informações gerais sobre o objetivo da pesquisa e instruções de como participar de sua realização.

Figura 27 – Tela de introdução à pesquisa

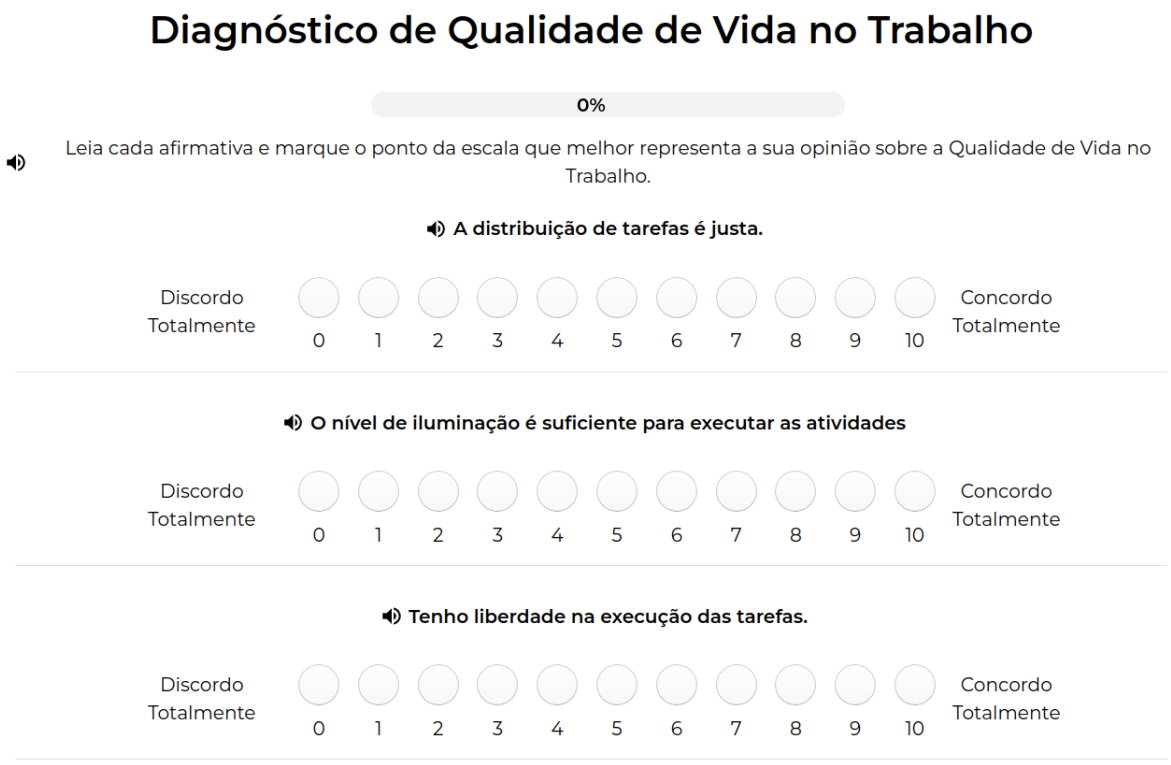
Fonte: Os próprios autores (2022a)

Ao confirmar estar de acordo em participar da pesquisa, o usuário deve inserir seu código de acesso gerado por um dos gestores na aba de pesquisas e assim começar a responder o questionário.

6.4 Questionário

O questionário é composto por questões de natureza quantitativa (escala psicométrica do tipo *Likert* e qualitativa (questões abertas) que permitem conhecer, com rigor científico, o que pensam os respondentes sobre a Qualidade de Vida no Trabalho em uma dada organização (FERREIRA, 2009). A Figura 28 mostra um exemplo de como é mostrado o questionário para o usuário respondente.

Figura 28 – Tela do questionário



Fonte: Os próprios autores (2022a)

Ao finalizar o questionário o usuário recebe uma mensagem de agradecimento personalizada para que tenha ciência do fim da atividade. Com isso, os principais fluxos do produto foram finalizados.

7 Conclusão

O projeto foi constituído de duas etapas, sendo a primeira de planejamento e especificação e a segunda de desenvolvimento do produto mínimo viável. Além das duas etapas citadas, foram criadas histórias de usuário para o período pós-MVP, mas que não fazem parte do escopo desse projeto.

A primeira etapa foi finalizada durante o TCC1, no qual a ideia do produto foi preparada para o desenvolvimento. Já no TCC2, a equipe utilizou dos artefatos gerados durante as etapas anteriores para desenvolver o produto. Portanto, após finalizar as duas etapas, o time finalizou o processo de criação do MVP.

A equipe passou pelas principais etapas da construção de um produto de *software*, sendo elas: alinhamento de visão com o cliente, criação do *backlog*, documentação técnica, preparação do ambiente e desenvolvimento do produto.

Foram criados diversos artefatos de documentação para auxiliar o desenvolvimento do produto, desde os artefatos da *Lean Inception*, detalhados no Capítulo 3 focados no alinhamento da visão do produto, até o documento de arquitetura presente no Apêndice A que contém informações técnicas sobre modelagem e divisões dos microsserviços.

Após a preparação para o desenvolvimento do produto, o time trabalhou nas histórias de usuário definidas na etapa de criação do *backlog*. Houveram contratempos pelo fato da equipe de desenvolvimento ser pequena, mas todos foram relatados ao cliente, mantendo assim as expectativas alinhadas. As histórias finalizadas podem ser vistas na Tabela 14.

Tabela 14 – Histórias de usuário finalizadas

US	**Nome**
US01	Acesso ao questionário
US02	Entrada manual de informações do GT
US07	Customização de formulários
US08	Criação de usuários do tipo administrador
US09	Criação de usuários do tipo gestor
US10	*Login* na plataforma
US12	Criação termo de compromisso
US14	Formulário base para a coleta
US17	Mensagem de confirmação de envio
US18	Alteração de tamanho da fonte
US19	Navegação do formulário pelo teclado

US20	Botão de leitura de pergunta
US21	Aplicação de técnicas estatísticas para cada tipo de pergunta
US24	Exportação de dados
US25	Geração de gráficos base
US27	Filtrar dados
US30	Visualização de respostas

Fonte: Os próprios autores (2022e)

As histórias de usuário de maior importância para o uso do produto foram finalizadas, com base em reuniões periódicas com o cliente. No total foram desenvolvidas 17 histórias de usuário para o MVP e 30 foram planejadas para o produto completo. Com isso a etapa final do escopo do projeto foi finalizada.

Portanto são considerados concluídos os objetivos específicos definidos neste trabalho (vide Seção 1.1) sendo o primeiro de identificação dos requisitos realizado na primeira parte deste trabalho e os outros dois de desenvolvimento de: uma solução capaz de coletar, tratar, armazenar e visualizar os dados da aplicação, e um sistema que garanta a confidencialidade das informações dos usuários durante a etapa da coleta de dados, concluídos na segunda etapa como planejado pelos autores.

7.1 Importância do projeto para a formação

O desenvolvimento do projeto foi de extrema importância para a formação dos estudantes. Além de permitir que os alunos pudessem colocar em prática os conhecimentos teóricos adquiridos durante o curso, foi dada a oportunidade de fazer parte de um projeto que tem o intuito de beneficiar a sociedade.

Para desenvolver o projeto foram colocados em prática conhecimentos teóricos de diversas áreas de estudo do curso de Engenharia de *Software*, como: lógica de programação, elaboração de requisitos, métodos de desenvolvimento de *software*, qualidade e arquitetura de *software* e sistemas de banco de dados.

De todas as áreas colocadas em prática durante o projeto, as que o time mais ganhou experiência foram as de elaboração de requisitos e os diferentes métodos de *software*. O desenvolvimento do produto foi extremamente importante, mas o time estava preparado, visto que a maioria da tecnologia utilizada já era de conhecimento dos desenvolvedores.

Utilizar métodos de criação de um produto em conjunto com o cliente é uma atividade normalmente reservada ao âmbito profissional, mas que foi possível realizar

no projeto. Com isso o time pode ganhar mais experiência nessa área, melhorando as habilidades técnicas, como o gerenciamento de um projeto de *software*, e interpessoais, como a comunicação real com o cliente.

7.2 Trabalhos futuros

O resultado desse TCC foi o MVP, ou produto mínimo viável. Portanto, ele ainda possui funcionalidades ainda não implementadas, mas não importantes para o seu funcionamento. A lista de funcionalidades que não foram incluídas no produto desenvolvido por esse TCC é apresentada na Tabela 15.

Tabela 15 – Histórias de usuário para desenvolvimento futuro

US	**Nome**
US03	*Upload* de lista com informações do GT
US04	Lembretes
US06	Envio de mensagens para o GT
US11	Criação de novas versões do formulário
US13	*Log* de operações dos gestores
US15	Gravação de tempo de resposta
US16	Mecanismo para verificação de inatividade
US22	Aplicação de outras técnicas estatísticas
US23	Aplicação técnicas estatísticas para detectar respostas inadequadas
US26	Geração de gráficos adicionais
US28	Geração de relatórios intermediários
US29	Geração de relatório final

Fonte: Os próprios autores (2022e)

Portanto, como trabalho futuro, é esperado que sejam desenvolvidas as histórias de usuário citadas na Tabela 15.

Apêndices

APÊNDICE A – Documento de Arquitetura

A.1 Introdução

Este documento tem como objetivo apresentar uma visão geral abrangente da arquitetura de software do sistema e especificar decisões arquiteturais pertinentes ao desenvolvimento da aplicação IA_QVT. Nele estão contidos os seguintes pontos respectivamente: Representação da Arquitetura, Metas e restrições de Arquitetura, Visão lógica e Visão de implementação.

A.1.1 Finalidade

Este documento tem como objetivo apresentar uma visão geral abrangente da arquitetura de software do sistema e especificar decisões arquiteturais pertinentes ao desenvolvimento da aplicação IA_QVT(Inventário Avaliativo de Qualidade de Vida no Trabalho).

A.1.2 Escopo

O IA_QVT é uma plataforma Web projetada para permitir a aplicação do questionário sobre a Qualidade de Vida no Trabalho (QVT) e obtenção de estatísticas e relatórios dos mesmos. Os desenvolvedores são responsáveis por seguir este documento, visando garantir o padrão proposto para a arquitetura.

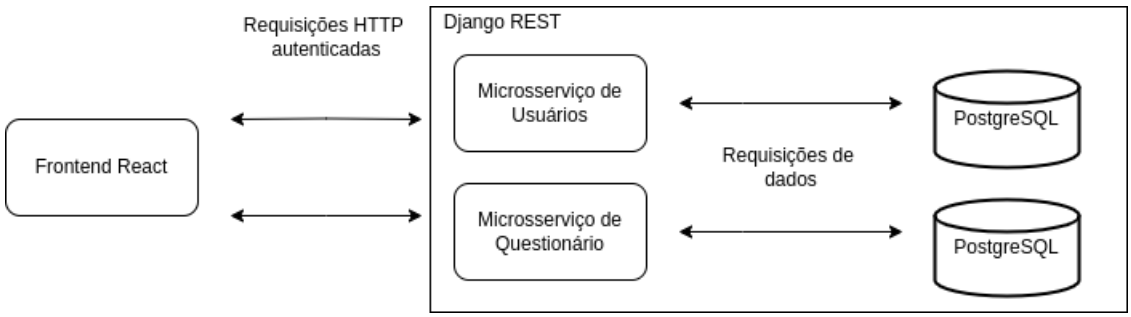
A.1.3 Visão Geral

Este documento contém os detalhes sobre as características arquiteturais escolhidas pela equipe de desenvolvimento para a solução em software do projeto IA_QVT.

A.2 Representação da Arquitetura

A arquitetura utilizada no projeto será baseada em microsserviços. Microsserviço é uma abordagem para desenvolver uma única aplicação como um conjunto de serviços, cada um executado em seu próprio processo e se comunicando através de mecanismos leves, geralmente, através do protocolo HTTP. Estes serviços são publicados em produção de maneira independente e o digrama de relações presente na Figura 29 mostra, de maneira geral, o seu funcionamento.

Figura 29 – Diagrama de Relações



Fonte: Rossi e Rocha (2022c)

A.3 Metas e Restrições de Arquitetura

As linguagens que serão utilizadas são: *JavaScript* no *frontend*, para o desenvolvimento da Interface de interação com o usuário com auxílio da biblioteca *React*, e *Python* no *backend*, para criação da API responsável pela comunicação entre os serviços da aplicação. A interface será desenvolvida como plataforma *Web*, apenas em português brasileiro.

A.4 Visão lógica

Descreve como o sistema é estruturado, em termos de unidades de implementação. Os elementos são pacotes, classes e interfaces. O relacionamento entre os elementos mostra as dependências, as realizações de interface e seus relacionamentos.

A.4.1 Visão geral: Pacotes e Camadas

A aplicação IA_QVT será construída sobre a biblioteca *React* no *frontend* e sobre o *Django REST Framework* no *backend*. O *React* é uma biblioteca responsável apenas pela parte da *view* apesar de que com a adição de alguns outros componentes podemos encaixá-lo em uma arquitetura MVC. O *framework Django* utiliza uma variação da arquitetura MVC, chamada MTV(Model Template View) onde a *view* delega a informação apresentada para o *template*. O desenvolvimento em camadas e componentes é benéfico, pois tem custos reduzidos de desenvolvimento e manutenção e, também, é de fácil reutilização em outros projetos.

A.4.1.1 *Model*

A *model* é uma interface de apresentação do banco de dados que permite o uso de seus dados sem conhecer suas complexidades. O modelo também fornece uma camada

de abstração, tornando possível o uso do mesmo modelo com vários tipos de bancos de dados.

A.4.1.2 *View*

Estabelece a conexão entre a *Model* e o *Template*. Ela recebe as requisições do usuário, acessa o banco de dados e retorna a informação solicitada.

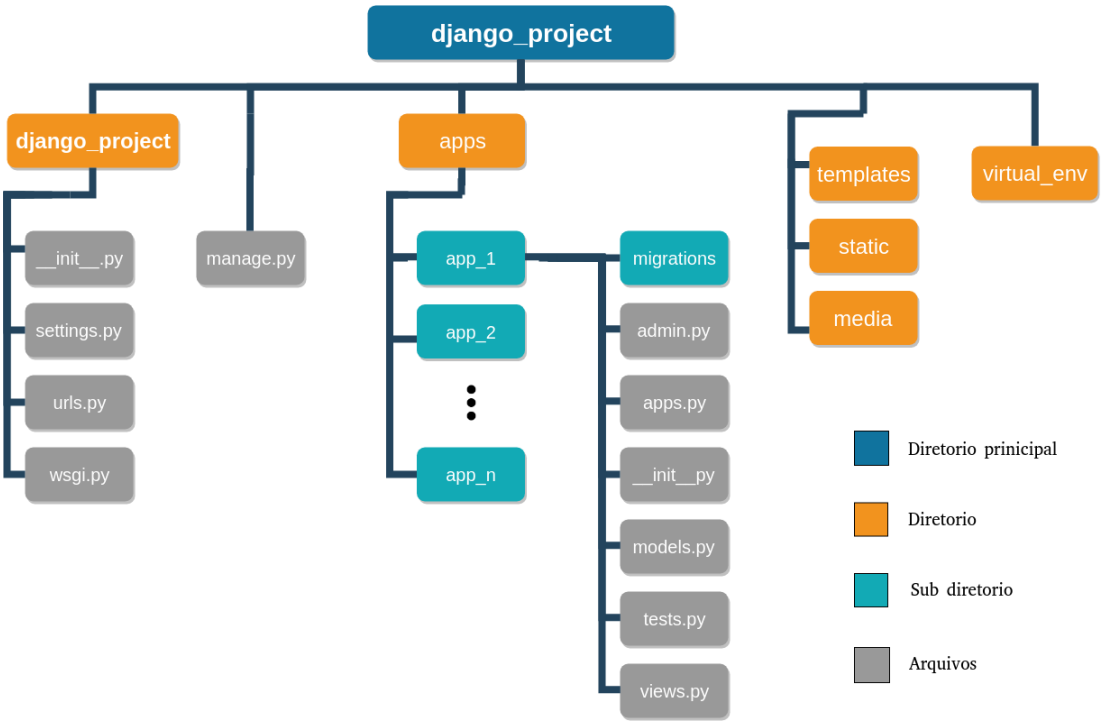
A.4.1.3 *Template*

É a camada de apresentação onde se decide como alguma informação do banco de dados deve ser apresentada para o usuário.

A.4.1.4 *Django Rest Framework*

O *Django REST framework* organiza o projeto em diretórios, em que cada um contém uma funcionalidade independente do restante da aplicação, como sugere a Figura 30.

Figura 30 – Estrutura de diretórios Django



Fonte: (SAMYAK, 2022)

- *apps*: cada aplicação tem uma pasta com as suas *models*, *views*, formulários, testes, *templates* e arquivos estáticos. Além disso, também há um arquivo *URLs* que será incluso no *URLs* global.

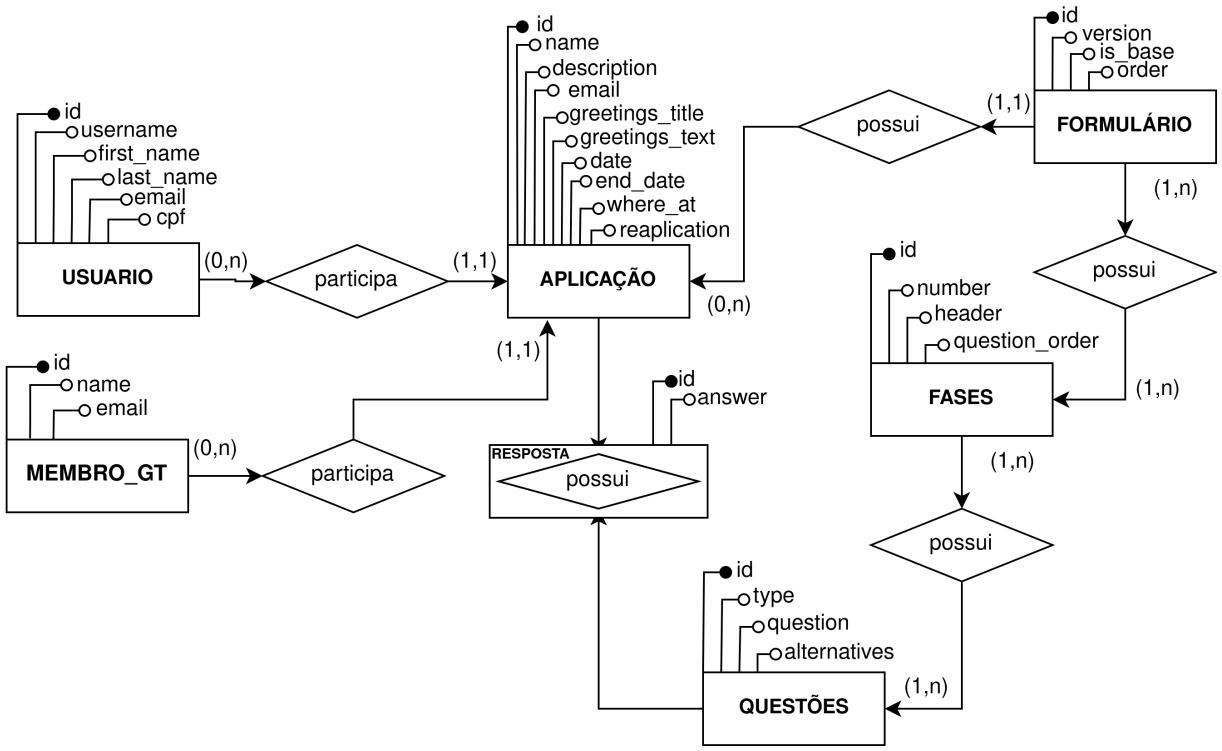
- *migrations*: pasta com as migrações para o banco de dados;
- *static*: pasta com arquivos *JavaScript*, *CSS* e imagens;
- *admin*: arquivo de conexão do *app* com o *admin*;
- *templates*: arquivos *html* do projeto;
- *virtual_env*: isolar o ambiente do resto do computador;
- *tests.py*: arquivos de testes referente ao *app*;
- *models.py*:arquivos de *models* do *app*;
- *views.py*: arquivos de *views* do *app*;
- *urls.py*: arquivo que mapeia as *views* com os *templates* de cada *app*;
- *__init__.py*: arquivo que transforma as configurações em um pacote *python*;
- *settings.py*: arquivos com as configurações básicas da aplicação;
- *wsgi.py*: especificação para uma interface simples e universal entre servidores *web* e aplicações *web*;
- *manage.py*: arquivo criado automaticamente pelo *Django REST* para gerenciamento de comandos.

A.5 Visão de Implementação

A.5.1 Diagrama Entidade Relacionamento

Um diagrama entidade relacionamento (DER) é um tipo de fluxograma que ilustra como “entidades” - pessoas, objetos ou conceitos - se relacionam entre si dentro de um sistema. Nosso diagrama está representado pela Figura 31 e nele estão representados cinco entidades: Usuário, Permissão, Questão, Formulário e Aplicação. No diagrama, podemos perceber que a entidade Aplicação é a entidade principal do projeto, pois é nela que se concentram a maior parte dos relacionamentos.

Figura 31 – Diagrama Entidade Relacionamento



Fonte: Rossi e Rocha (2022d)

A.5.2 Tamanho e desempenho

O sistema é uma aplicação Web, no qual os principais objetivos são permitir ao usuário responder o questionário sobre QVT, gerar estatísticas sobre as respostas e permitir que usuários gestores possam realizar modificações e acompanhamento dos questionários.

É esperado que o software seja utilizado por três tipos de usuários: o administrador, o gestor e o usuário respondente. O usuário administrador poderá adicionar novos usuários gestores e terá acesso a todas funcionalidade da aplicação. O usuário gestor terá acesso às estatísticas geradas a partir do questionário e também poderá modificar o questionário para se adequar a necessidade do local onde ele será aplicado. Já o usuário respondente terá acesso apenas ao questionário de forma anônima e deverá respondê-lo.

Para o microsserviço de usuário não será necessário um banco de dados muito robusto, tendo em vista que existirão poucos usuários. Porém, para o banco de dados responsável por armazenar os dados dos questionários torna-se necessário um banco um pouco mais robusto, levando em consideração que serão armazenados dados de vários respondentes diferentes, o que implica que seja utilizado um serviço de *Cloud Server*.

Plataformas mais simples como *Heroku* poderão ser usadas durante o tempo de adesão da aplicação, porém, quando em pleno funcionamento, uma plataforma mais po-

tente pode ser mais adequada para atender às demandas do cliente. A arquitetura foi escolhida de forma que a aplicação tenha aparatos de armazenamento, busca, e visualização de dados suficientemente eficientes para que possam atender, de forma satisfatória, até aparelhos celulares.

Referências

AHMAD, M. O.; MARKKULA, J.; OIVO, M. Kanban in software development: A systematic literature review. 2013. Disponível em: <https://www.researchgate.net/publication/260739586_Kanban_in_Software_Development_A_Systematic_Literature_Review>. Citado na página 23.

CAROLLI, P. *Lean Inception: How to align people and build the right product*. [S.l.]: Editora Carolli, 2018. Citado 2 vezes nas páginas 19 e 20.

CAROLLI, P. Scrum starts with pbb. 2019. Disponível em: <https://www.caroli.org/en/scrum-starts-with-pbb/>. Citado na página 21.

COHN, M. *Why the Fibonacci Sequence Works Well for Estimating*. 2019. Acessado em 04 de Junho de 2022. Disponível em: <https://www.mountaingoatsoftware.com/blog/why-the-fibonacci-sequence-works-well-for-estimating>. Citado na página 21.

CONTRIBUTORS, M. *JavaScript*. 2021. Acessado em 1 de Março de 2022. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Citado na página 27.

COUTINHO, T. *Sprint Scrum: o que é e como funciona?* 2018. Acessado em 12 de Abril de 2022. Disponível em: <https://www.voitto.com.br/blog/artigo/sprint-scrum>. Citado na página 57.

DOCKER. *Overview of Docker Compose*. 2022. Acessado em 24 de Fevereiro de 2022. Disponível em: <https://docs.docker.com/compose/>. Citado na página 28.

DOCKER. *Use containers to Build, Share and Run your applications*. 2022. Acessado em 24 de Fevereiro de 2022. Disponível em: <https://www.docker.com/resources/what-container>. Citado na página 28.

DRAGONI, N. et al. Microservices: yesterday, today, and tomorrow. 2017. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-319-67425-4_12>. Citado 2 vezes nas páginas 55 e 56.

FERREIRA, M. C. *Inventário de Avaliação de Qualidade de Vida no Trabalho*. 2009. Acessado em 05 de Setembro de 2022. Disponível em: <http://www.ergopublic.com.br/?pg=textos_gerais&id=4>. Citado na página 72.

FERREIRA, M. C. A ergonomia da atividade pode promover a qualidade de vida no trabalho? reflexões de natureza metodológica. 2011. Disponível em: <https://periodicos.ufsc.br/index.php/rpot/article/view/22243>. Citado na página 18.

FERREIRA, M. C. *Qualidade de vida no trabalho: uma abordagem centrada no olhar dos trabalhadores*. [S.l.]: Editora Paralelo 15, 2017. Citado na página 15.

FERREIRA, M. C. *Diagnóstico, Política e Programa de Qualidade de Vida no Trabalho (QVT)*. [S.l.]: Editora Brazil Publishing, 2019. Citado na página 15.

GIT. *About*. 2022. Acessado em 21 de Fevereiro de 2022. Disponível em: <https://git-scm.com/about/branching-and-merging>. Citado na página 28.

HORST, D. J. et al. Quality of working life and productivity: An overview of the conceptual framework. 2018. Disponível em: <https://www.researchgate.net/publication/338633211_A_NEW_WAY_OF_MANAGEMENT_A_SCRUM_MANAGEMENT>. Citado na página 15.

ISO/IEC 25010. *ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. 2011. Citado 2 vezes nas páginas 25 e 26.

ISO/IEC 25020. *ISO/IEC 25020:2019 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality measurement framework*. 2019. Citado na página 25.

KARABULUT, A.; ERGUN, E. A new way of management: A scrum management. 2014. Disponível em: <https://www.researchgate.net/publication/275344546_Quality_of_Working_Life_and_Productivity_An_Overview_of_the_Conceptual_Framework>. Citado na página 22.

PARADIGM, V. *What is Story Point in Agile? How to Estimate a User Story?* 2022. Acessado em 28 de Março de 2022. Disponível em: <https://www.visual-paradigm.com/scrum/what-is-story-point-in-agile/>. Citado na página 22.

PARADIGM, V. *What is User Story?* 2022. Acessado em 23 de Março de 2022. Disponível em: <https://www.visual-paradigm.com/guide/agile-software-development/what-is-user-story/>. Citado na página 21.

POSTGRESQL. *PostgreSQL*. 2022. Acessado em 21 de Março de 2022. Disponível em: <https://www.postgresql.org/>. Citado na página 27.

PRESSMAN, R. S.; MAXIM, B. R. *Software Engineering A PRACTITIONERS APPROACH*. [S.l.]: McGraw-Hill Education, 2015. Citado 2 vezes nas páginas 23 e 24.

PYTHON. *The Python Tutorial*. 2022. Acessado em 20 de Fevereiro de 2022. Disponível em: <https://docs.python.org/3/tutorial/index.html#tutorial-index>. Citado na página 27.

ROCHA, C.; NERI, H. *Gestão de Portfólios e Projetos de Software: Fases ou Grupos de Processo: Planejamento II*. 2016. Acessado em 15 de Agosto de 2022. Disponível em: <https://github.com/fga-eps-mds/A-Disciplina-MDS-EPS/blob/master/GPP_Material/06%20-%20Planejamento%20II/main.pdf>. Citado na página 54.

ROSSI, J.; ROCHA, P. *Aplicação IA-QVT*. 2022. Citado 10 vezes nas páginas 64, 65, 66, 67, 68, 69, 70, 71, 72 e 73.

ROSSI, J.; ROCHA, P. *Diagrama de etapas TCC IA QVT*. 2022. Citado na página 29.

ROSSI, J.; ROCHA, P. *Diagrama de relações*. 2022. Citado na página 82.

ROSSI, J.; ROCHA, P. *Diagrama Entidade Relacionamento*. 2022. Citado 2 vezes nas páginas 59 e 85.

ROSSI, J.; ROCHA, P. *Modelo de processos TCC IA QVT*. 2022. Citado 9 vezes nas páginas 19, 31, 32, 48, 49, 50, 51, 76 e 77.

ROSSI, J.; ROCHA, P. *Sprint Backlog*. 2022. Citado 5 vezes nas páginas 57, 58, 59, 60 e 61.

ROSSI, J. et al. *Lean Inception IA QVT*. 2022. Disponível em: <https://www.figma.com/file/NpoDM4RXmGjvoYXZEVndKB/Lean-Inception-TCC-IA-QVT>. Citado 6 vezes nas páginas 36, 37, 38, 39, 40 e 55.

ROSSI, J. et al. *PBB IA QVT*. 2022. Disponível em: <https://app.mural.co/t/tcciaqvt4561/m/tcciaqvt4561/1647013533650/1246ce7f1a006217ce677e2cbeaea3e0c01b6c61?sender=u4279aac9c4b5e91d70730482>. Citado 5 vezes nas páginas 42, 43, 44, 45 e 46.

SAMYAK. *Django Project Structure Best Practice 2022 - Django Tutorial*. 2022. Disponível em: <https://studygyaan.com/django/best-practice-to-structure-django-project-directories-and-files>. Citado na página 83.

SOMMERVILLE, I. *Software Engineering 9th edition*. [S.l.]: Addison-Wesley, 2011. Citado na página 23.

SOURCE, F. O. *React: A JavaScript library for building user interfaces*. 2022. Acessado em 1 de Março de 2022. Disponível em: <https://reactjs.org>. Citado na página 27.

WALTON, R. E. *Quality of Working Life: What Is It?* [S.l.]: Sloan Management Review, 1973. Citado na página 17.

ZENHUB. *Our Story*. 2022. Acessado em 21 de Fevereiro de 2022. Disponível em: <https://www.zenhub.com/about-us>. Citado na página 28.