Corigge

VISÃO DO PRODUTO E PROJETO

Versão 1.2

Histórico de Revisão

Data	Versão	Descrição	Autor
05/04/2025	0.1	Completando seções 1 e 2	Otávio
08/04/2025	0.2	Corrigindo erros apontados na versão 0.1	Otávio
21/04/2025	1.0	Adicionando seções 4,5 e 6	Todos
21/04/2025	1.1	Corrigindo seções 1 e 2 com base em feedback do monitor	Otavio
26/05/2025	1.2	Adcionando seções 4, 7, 8, 9 e 10	Todos

1 Cenário Atual do Cliente e do Negócio

1.1 Introdução ao Negócio e Contexto

A Guia do PAS é uma empresa especializada na preparação de estudantes para o Programa de Avaliação Seriada (PAS) da Universidade de Brasília (UnB), um dos processos seletivos mais concorridos do país. Atuando no setor educacional, a empresa oferece conteúdos direcionados, simulados, correções comentadas, aulas ao vivo e materiais exclusivos, com foco na aprovação dos alunos. A empresa, nos últimos anos, também tem expandido seu mercado para o abranger o atendimento à outras instituições de ensino, oferecendo a aplicação de simulados, listas de exercícios personalizadas e controle de desempenho dos alunos. Além disso, a empresa tem buscado expandir sua atuação para outros processos seletivos além do PAS, visando aumentar sua base de clientes e diversificar seus serviços.

1.2 Identificação da Oportunidade ou Problema

Com o crescimento da base de usuários e o aumento no número de clientes B2B buscando a realização de simulados em suas escolas, percebeu-se algumas dificuldades na obtenção das respostas dos alunos, na inserção desses dados no banco de dados da empresa e a visualização desses dados de forma efetiva pelos coordenadores das escolas parceiras. Após a aplicação do simulado é necessário registrar as respostas de cada aluno e associar esses dados com a matrícula do aluno, utilizando desses dados em uma planilha .csv para ser cadastrada no sistema da empresa. Mesmo fazendo uso de serviços de outras empresas para realizar essa identificação dos gabaritos, essas soluções eram demoradas e muito custosas (demorando mais de um mês para receber os resultados e custando em média R$1500.00 por simulado), não tendo sido possível encontrar uma solução rápida, de fácil uso e barata para a realização de grandes simulados ou de simulados em várias escolas em um curto período de tempo.

A empresa já tentou desenvolver uma solução anteriormente, também chamada Corigge, mas esta primeira versão apresentou diversos problemas técnicos que comprometeram sua utilização efetiva. O sistema anterior, que era uma aplicação web, necessitava de manutenção constante e apresentava várias limitações operacionais: o servidor precisava estar sempre ativo em um computador específico com boa capacidade de processamento, onde um script Python precisava ser executado manualmente para permitir que outros usuários acessassem o sistema. Além disso, o sistema apresentava instabilidades frequentes, com múltiplas ocorrências de crashes tanto no servidor quanto na aplicação, sem mecanismos de auto-recuperação. A necessidade de ajustes manuais de parâmetros pelos usuários finais também se mostrou um obstáculo significativo para a adoção do sistema. A funcionalidade do sistema anterior era limitada apenas à identificação das respostas dos alunos, sem oferecer recursos adicionais como análise de desempenho, geração de relatórios ou personalização de gabaritos.

Diante dessas experiências anteriores e da necessidade de uma solução mais robusta, a empresa busca o desenvolvimento de uma nova versão do sistema como uma aplicação desktop, que permita a correção automática dos gabaritos de forma eficiente e rápida, desejando também oferecer esse serviço para outras instituições de ensino não associadas à empresa.

Figura 1: Diagrama de Ishikawa (Fishbone) com os principais desafios do projeto

1.3 Desafios do Projeto

Os principais desafios do projeto são: A utilização e parametrização de ferramentas de CV (Visão computacional) para a identificação das respostas dos alunos, a não padronização dos gabaritos preenchidos pelos alunos, a não familiaridade de alguns membros da equipe com as ferramentas a serem utilizadas, a criação de um design amigável, eficiente e intuitivo para a utilização pelos membros da empresa, e o empacotamento das ferramentas necessárias para integrar o CV em uma aplicação multiplataforma em flutter.

1.4 Segmentação de Clientes

O público alvo atual da empresa são alunos do ensino médio e alunos de cursinhos que buscam uma preparação estratégica e eficiente para o PAS, além de instituições de ensino que desejam ter uma preparação mais específica para o PAS e simulados personalizados para esta preparação.

O perfil dos alunos é de jovens entre 15 e 20 anos, que buscam uma preparação diferenciada para o PAS, com foco em resultados e aprovação. Eles valorizam a qualidade do conteúdo, a eficiência dos simulados e a possibilidade de acompanhar seu desempenho ao longo do tempo.

O perfil dos coordenadores das instituições de ensino são profissionais com formação superior, que atuam na área de educação e buscam soluções tecnológicas para otimizar a gestão acadêmica e melhorar a experiência dos alunos. Eles valorizam a eficiência e a facilidade de uso das ferramentas que utilizam.

2 Solução Proposta

2.1 Objetivos do Produto

O objetivo principal do produto é otimizar o processo de correção de gabaritos da Guia do PAS, reduzindo o tempo de processamento de 1 mês para menos de 24 horas, diminuindo o custo por correção de R$5,00 para menos de R$1,00, e aumentando a precisão da identificação das respostas para mais de 99%. Além disso, o sistema deve permitir a análise automática do desempenho individual e coletivo dos alunos, gerando relatórios detalhados para o gestor que incluem estatísticas por questão, comparações entre turmas e identificação de padrões de erro e relatórios individuais para os alunos com a nota do aluno, a posição dele entre os outros alunos que fizeram a prova e a nota média da turma. O produto também visa expandir a base de clientes da empresa, permitindo a oferta do serviço de correção para outras instituições de ensino, com meta de atingir pelo menos 10 novas escolas parceiras no primeiro ano de operação. Por fim, o sistema deve resolver os problemas técnicos encontrados na versão anterior, como instabilidades, necessidade de manutenção constante e limitações operacionais, oferecendo uma solução robusta e multiplataforma que não dependa de servidores dedicados.

2.2 Características da Solução

O produto será um aplicativo desktop multiplataforma (Windows, Linux e MacOS) que deve ser capaz de processar imagens de gabaritos preenchidos pelos alunos, identificar as respostas dadas pelos alunos em cada questão, identificar a matrícula do aluno, comparar as respostas com o gabarito correto, gerar relatórios detalhados (com o desempenho do aluno — incluindo notas, acertos e erros —, comparações entre alunos, comparações entre grupos de alunos e dados estatísticos de cada questão). O produto também deve ser capaz de gerar templates de gabaritos e permitir a personalização dos gabaritos de acordo com as necessidades de cada instituição de ensino. Além disso, o produto também deve permitir a exportação dos dados e relatórios para arquivos .csv e .pdf, para facilitar a integração com outros sistemas e a análise dos dados. A escolha por uma aplicação desktop também visa facilitar o processo de desenvolvimento, permitindo que a equipe de desenvolvimento da empresa adicione mais funcionalidades ao longo do tempo de forma ágil e eficiente.

2.3 Tecnologias a Serem Utilizadas

As tecnologias a serem utilizadas para o desenvolvimento do produto são:

- Frontend: Dart + Flutter
- Identificação de imagem local: Python + OpenCV
- Backend: Typescript + Express.js
- Banco de dados: Supabase (PostgreSQL)
- Hospedagem: Azure + VM Linux

2.4 Pesquisa de Mercado e Análise Competitiva

- **RemarkOffice**: Possui um preço muito alto ($1,195.00), não oferece a possibilidade de personalização dos gabaritos e não possui precificação por gabarito analisado (deixando o serviço mais barato para pequenas empresas).
- **Gradepen**: É gratuito mas pouco conhecido. Também não oferece estatísticas de desempenho do aluno, apenas a correção do gabarito. Também não oferece a possibilidade de personalização dos gabaritos.
- **Prova Fácil**: Custa $0.15 por correção, não oferece estatísticas de desempenho do aluno, apenas a correção do gabarito. Também não oferece a possibilidade de personalização dos gabaritos.

2.5 Análise de Viabilidade

A viabilidade técnica do projeto é alta, uma vez que a equipe possui experiência em desenvolvimento de plataformas similares e familiaridade com as tecnologias propostas como **Flutter**, **PostgreSQL**, **Typescript** e a com os **serviços de cloud** a serem utilizados. O prazo é de 3 meses para o desevolvimento da solução, o que é viável considerando a complexidade do projeto e a experiência da equipe. O custo estimado para o desenvolvimento é de R$1,000.00 para eventuais pagamentos de serviços como o banco de dados e hosteamento dos servidores backend durante o desenvolvimento, o que é viável considerando a demanda do mercado e o preço cobrado pelos concorrentes. O retorno financeiro inicial esperado é de R$1,000.00 por mês (uma estimativa baixa considerando apenas 2000 correções por mês à R$0.50 por correção, o que é facilmente atingível dado o crescimento atual da empresa no mercado de Brasília e a possibilidade de oferecimento desse serviço para diversas empresas de educação do país), considerando a venda do serviço para outras instituições de ensino e o crescimento da confiabilidade na empresa, o que permite o oferecimento do serviço para várias instituições de Brasília.

2.6 Impacto da Solução

A solução proposta terá um impacto significativo na eficiência operacional da empresa, reduzindo o tempo (de um mês para algumas horas) e os custos associados (de R$5,00 para menos de R$1,00 por correção) à correção de gabaritos. Além disso, a solução permitirá a venda desse serviço para outras instituições de ensino, aumentando a receita da empresa em pelo menos R$12.000,00 por ano e expandindo sua atuação no mercado educacional, expandindo a influência da empresa no mercado de preparação para o PAS. A solução também melhorará a experiência do usuário, proporcionando resultados mais rápidos (análise de 100 gabaritos por minuto) e precisos, além de relatórios detalhados sobre o desempenho dos alunos.

3 Estratégias de Engenharia de Software

3.1 Estratégia Priorizada

- **Abordagem de Desenvolvimento de Software:** Híbrido (Ágil com elementos dirigidos por plano).

- **Ciclo de vida:** Iterativo e incremental
- **Processo de Engenharia de Software:** OpenUP.

O OpenUP será utilizado, pois é um processo ágil e leve, adequado para projetos de médio porte com necessidades de validação rápida e evoluções constantes.

As quatro fases do OpenUP serão seguidas:

- **Concepção:** Definição dos objetivos do projeto, levantamento dos requisitos iniciais, entendimento do problema de correção de gabaritos, escolha das tecnologias (Flutter, OpenCV, Typescript, Supabase) e planejamento inicial de prazos e custos.
- **Elaboração:** Refinamento dos requisitos (como detalhamento das funcionalidades de correção, exportação e geração de relatórios), desenvolvimento de protótipos iniciais de reconhecimento de gabaritos, e avaliação dos principais riscos (como o reconhecimento de imagens e integração entre as tecnologias).
- **Construção:** Desenvolvimento iterativo da aplicação desktop, integração contínua entre frontend, backend e sistemas de visão computacional, testes de funcionalidades como reconhecimento automático e geração de relatórios, além de ajustes baseados no feedback interno.
- **Transição:** Testes finais de desempenho e robustez (ex: correção de 100 gabaritos por minuto), implantação para o ambiente de produção, produção de documentação de uso para a equipe comercial e suporte inicial aos clientes externos.

O cronograma de execução do projeto será alinhado às entregas previstas em cada uma dessas fases, respeitando o prazo de 3 meses para conclusão do produto mínimo viável (MVP).

3.2 Quadro Comparativo

Característica	Unified Process (UP)	Open Unified Process (OpenUP)
Objetivo	Metodologia robusta com forte foco em documentação e controle	Processo leve e ágil com foco em colaboração e entregas contínuas
Foco no Processo	Estruturado e formal, com controle rigoroso	Flexível e centrado na produtividade da equipe
Complexidade	Elevada, ideal para projetos críticos e de grande porte	Reduzida, ideal para projetos de médio porte que exigem agilidade
Fases	Concepção, Elaboração, Construção, Transição (com ampla documentação)	Concepção, Elaboração, Construção, Transição (com entregas enxutas)
Disciplinas	Abrange várias disciplinas como Modelagem, Design, Testes, Gerência de Projeto	Enfoque em Requisitos, Implementação, Testes e Gerência de Projeto
Documentação	Detalhada e exigida em cada fase	Essencial apenas para manter a produtividade da equipe
Iteratividade	Iterativo e incremental, com processos mais pesados	Iterativo e incremental, com ciclos curtos e adaptáveis
Gestão de Riscos	Formal e concentrada na fase de Elaboração	Contínua e integrada de forma leve
Flexibilidade	Menor adaptabilidade a mudanças rápidas	Alta flexibilidade, facilitando ajustes frequentes
Colaboração	Papéis bem definidos e estrutura hierárquica	Equipes multifuncionais, com papéis flexíveis e foco na colaboração
Recomendado para	Projetos grandes, críticos, com alta demanda por rastreabilidade e validações formais	Projetos médios ou pequenos, com necessidade de agilidade – como no projeto Corigge

3.3 Justificativa

O projeto da nova solução **Corigge** para a Guia do PAS demanda um processo ágil, adaptável e eficiente. O **OpenUP** se mostra a escolha ideal por:

- Permitir a adaptação rápida a mudanças no escopo, como ajustes em funcionalidades (ex: novos tipos de gabaritos ou estatísticas).
- Exigir menos documentação formal, favorecendo o foco em entregas práticas e ágeis, especialmente importante para uma equipe enxuta.

- Ser mais leve que o UP tradicional, o que reduz a carga de burocracia, otimiza o tempo e diminui o custo do desenvolvimento — aspectos cruciais para um projeto com orçamento de aproximadamente R$ 1.000,00.
- Permitir iterações rápidas e entregas frequentes, essenciais para testar rapidamente o reconhecimento de imagens, relatórios e exportações, validando a qualidade da solução antes do lançamento comercial.
- Promover a colaboração ativa da equipe de desenvolvimento com os stakeholders (Guia do PAS e instituições parceiras), para garantir que o produto atenda às expectativas tanto para uso interno quanto para venda externa.

Em resumo, o **OpenUP** encaixa-se perfeitamente nas necessidades do projeto Corigge: é leve, iterativo, adaptável, reduz riscos e acelera o retorno sobre o investimento.

4 Engenharia de Requisitos

4.1 Atividades e Técnicas de ER

- **Elicitação e Descoberta**
  - **Entrevista com o cliente**: Realizar entrevistas para entender as necessidades, expectativas e objetivos principais do projeto, garantindo que os requisitos reflitam as prioridades do cliente.
  - **Análise de concorrentes**: Estudo detalhado das soluções existentes no mercado para identificar boas práticas e oportunidades de diferenciação.
  - **Brainstorming**: Sessões de brainstorming com a equipe para levantar ideias e explorar soluções inovadoras para atender às necessidades do cliente.

- **Análise e Consenso**
  - **Análise de Domínio de Requisito**: Entendimento aprofundado das áreas envolvidas no projeto, assegurando que os requisitos sejam viáveis e relevantes ao contexto do cliente.
  - **Análise de viabilidade**: Avaliação da viabilidade técnica e temporal dos requisitos levantados.
  - **Reuniões entre os membros da equipe**: Discussão e alinhamento dos objetivos e prioridades, promovendo o consenso entre os participantes.

- **Declaração de Requisitos**
  - **Reuniões entre os membros da equipe**: Sessões colaborativas para redigir e validar os documentos de requisitos.
  - **Documento de visão de produto**: Documento que descreve o escopo e as metas principais do projeto, alinhando todos os stakeholders.

- **Especificação de Requisitos**: Documento detalhado contendo os requisitos funcionais e não funcionais do sistema.
- **Features**: Listagem das principais funcionalidades organizadas em níveis hierárquicos (temas, épicos e histórias de usuário).

- **Verificação e Validação de Requisitos**
  - **Reuniões entre os membros da equipe**: Sessões colaborativas para verificar e validar os requisitos.
  - **Revisão dos critérios de aceitação**: Definir critérios objetivos para validar que os requisitos foram corretamente implementados.
  - **Brainstorming**: Sessões para validar ideias, explorar melhorias e alinhar expectativas sobre os requisitos levantados.

- **Organização e Atualização de Requisitos**
  - **MoSCoW**: Aplicação do método MoSCoW para priorização dos requisitos.
  - **User Story**: Abordagem para descrever os requisitos sob a perspectiva dos usuários.

- **Representação de Requisitos**
  - **Prototipagem**: Criação de protótipos para validar funcionalidades e melhorar a comunicação entre a equipe e os stakeholders.
  - **Diagramas**: Produção de diagramas (como arquitetura) para representar os processos e requisitos do sistema.
  - **Análise e Consenso**: Análise de risco — Identificação e análise de potenciais riscos ao projeto, incluindo mitigação de impactos e criação de planos de contingência.
  - **Verificação e Validação**: Walkthrough — Revisão com o cliente em cima do protótipo mostrando todas as funcionalidades que serão aplicadas ao produto.
  - **Feedback**: Feedback sobre o protótipo criado e melhorias que o cliente gostaria que fossem feitas.

- **Construção**
  - **Organização e Atualização**: Alinhamento da equipe — Realização de reuniões regulares para alinhar o progresso e ajustar as prioridades, garantindo que todos estejam na mesma direção.
  - **Feedback**: Coleta e aplicação de feedback para refinar as entregas e atender melhor às expectativas do cliente e da equipe.
  - **Verificação e Validação**: Entrevista com o cliente — Reuniões com o cliente para apresentar os resultados parciais e validar que os requisitos estão sendo atendidos.

  - **Feedback**: Recolher opiniões sobre o progresso e implementar melhorias com base nas observações recebidas.

- **Transição**

  - **Verificação e Validação**: Walkthrough — Sessões de revisão guiada onde as funcionalidades são apresentadas ao cliente e equipe para validação e ajustes finais.

4.2 Engenharia de Requisitos e o OpenUP

Fases do OpenUP	**Atividades da ER**	**Prática**	**Técnica**	**Resultados Esperados**
Concepção	Elicitação e Descoberta	Conhecimento do cliente e do problema	Entrevista com o cliente, Análise de Concorrentes, Brainstorming	Lista de necessidades, Declaração do problema, Lista de requisitos, Proposta de solução
Concepção	Análise e Consenso	Análise de requisitos	Análise de Domínio de Requisito, Análise de viabilidade, Reuniões entre a equipe	Criação do MVP
Concepção	Declaração de Requisitos	Registro dos requisitos	Reuniões da equipe, Documento de Visão do Produto, Especificação de Requisitos, Features	SRS - Software Requirements Specification
Concepção	Verificação e Validação de Requisitos	Validação de requisitos	Revisão de Critérios de Aceitação, Brainstorming, Reuniões da equipe	DoD e DoR
Concepção	Organização e Atualização	Priorização de requisitos	MoSCoW, User Story	Requisitos priorizados, Backlog de requisitos
Elaboração	Representação de Requisitos	Criação de protótipos	Prototipagem, Diagramas	Protótipo

Fases do OpenUP	Atividades da ER	Prática	Técnica	Resultados Esperados
Elaboração	Análise e Consenso	Alinhamento de requisitos	Análise de Risco, Lean Inception	User Story, SRS
Elaboração	Verificação e Validação	Validação do protótipo	Walkthrough, Feedback	Resultados do Walkthrough
Construção	Organização e Atualização	Revisão do produto	Alinhamento da equipe, Feedback	Atualização dos requisitos
Construção	Verificação e Validação	Revisão do produto	Walkthrough, Feedback, DoD	Resultados do Walkthrough
Transição	Verificação e Validação	Revisão do produto finalizado	Walkthrough	Resultados do Walkthrough, Qualidade dos Requisitos

5 Cronograma e Entregas

Fase	Sprint	Início	Fim	Objetivos e Entregas Esperadas
Concepção	Sprint 0	21/04/2025	08/05/2025	- Planejamento geral do projeto - Definição do escopo - Organização da documentação inicial - Definição das tecnologias principais - Definição do MVP do Corigge - Definição do DoR e DoD - Definição de User Stories - Declaração de requisitos funcionais e não funcionais - Ambiente de desenvolvimento inicial - Capacitação técnica (Flutter, OpenCV, Supabase) - Entrega da Unidade 1 em 22/04/2025 (terça-feira)
Elaboração	Sprint 1	09/05/2025	22/05/2025	- Protótipo no Figma - Definição da Arquitetura - Análise de riscos técnicos - Configuração do banco de dados (Supabase) - Workshop de Flutter - Workshop de OpenCV

Fase	Sprint	Início	Fim	Objetivos e Entregas Esperadas
Construção	Sprint 2	23/05/2025	05/06/2025	- Implementação inicial do frontend desktop, login e criação de conta - Implementação inicial da identificação das questões com OpenCV - Integração de frontend com a biblioteca python local - Criação dos endpoints backend para criação de conta - Personalização de gabaritos - Entrega da Unidade 2 em 27/05/2025 (terça-feira)
	Sprint 3	06/06/2025	19/06/2025	- Continuação da implementação do design no frontend - Testes de reconhecimento e comparação - Testes de criação de gabaritos - Melhorias de UI/UX - Ajustes baseados em feedback do cliente - Implementação inicial da geração de dashboards e relatórios
	Sprint 4	20/06/2025	03/07/2025	- Geração de relatórios simples (acertos e erros) para cada aluno - Implementação inicial de pagamentos com Stripe - Exportação de dados em .csv - Exportação de relatórios em .pdf - Validação das exportações - Entrega da Unidade 3 em 24/06/2025 (terça-feira)
Transição	Sprint 5	04/07/2025	17/07/2025	- Testes finais de performance (100 gabaritos/min) - Testes de estabilidade e recuperação - Melhorias de UI/UX - Ajustes baseados em feedback interno - Hospedagem do backend (Azure) - Walkthrough com o cliente (Guia do PAS) - Encerramento do projeto - Entrega da Unidade 4 em 15/07/2025 (terça-feira)

Datas específicas de entrega:

- Unidade 1 → 22/04/2025 (terça-feira)
- Unidade 2 → 27/05/2025 (terça-feira)
- Unidade 3 → 24/06/2025 (terça-feira)

- Unidade 4 → 15/07/2025 (terça-feira)

6 Interação entre Equipe e Cliente

6.1 Composição da Equipe

Papel	Descrição	Responsável	Participantes
Gerente de Projeto	Responsável por coordenar o projeto, manter a comunicação entre a equipe e o cliente, e garantir o cumprimento dos prazos e entregas.	Otavio Maya	-
Desenvolvedor FrontEnd	Cuida do desenvolvimento da interface do usuário, garantindo o design e a implementação das funcionalidades no lado do cliente.	Nathan Benigno	Otavio Maya
Desenvolvedor Backend	Desenvolve a lógica de negócio, realiza a integração com banco de dados e serviços externos por meio de APIs.	Esdras de Sousa	Atyrson Souto, Pedro Victor
Analista de Requisitos	Responsável por levantar, documentar e gerenciar requisitos funcionais e não funcionais, garantindo o alinhamento com os objetivos do projeto.	Eduardo, Nathan, Esdras, Otavio, Atyrson, Pedro e Adrian	Eduardo, Nathan, Esdras, Otavio, Atyrson, Pedro e Adrian
Analista de QA	Cria pipelines de integração e entrega contínua, desenvolve testes automatizados e realiza testes funcionais para assegurar a qualidade do produto.	Marcelo Adrian	-

Tabela 4: Composição da Equipe e Responsabilidades

6.2 Comunicação

6.2.1 Ferramentas Utilizadas

- **WhatsApp**: será utilizado para comunicações rápidas e diárias entre os membros da equipe e também com o cliente, sempre que necessário.
- **Google Meet**: usado para reuniões mais formais de sprint e também para os encontros de validação com o cliente.
- **GitHub Projects (Kanban)**: ferramenta usada para o acompanhamento visual das tarefas da equipe durante o desenvolvimento.

6.2.2 Tipos e Frequência das Reuniões

- **Revisão de Sprint** (a cada duas semanas):
  - Toda segunda-feira, a equipe fará uma reunião para revisar o que foi entregue na sprint anterior, identificando o que funcionou bem e o que pode ser melhorado.
- **Planejamento de Sprint** (a cada duas semanas):
  - Logo após a revisão, será feito o planejamento da nova sprint, com base no cronograma e nas prioridades do projeto.
- **Validação com o Cliente**:
  - As reuniões com o cliente ocorrerão **exclusivamente ao final das sprints**, alinhando-se à preferência do stakeholder por acompanhar apenas os fechamentos.
  - O objetivo dessas validações é apresentar o progresso alcançado, receber feedback e realizar ajustes, se necessário.
  - As reuniões acontecerão **preferencialmente às segundas ou sextas-feiras**, e não será exigida a presença de toda a equipe — apenas os membros responsáveis pelas entregas em validação participarão para conduzir a apresentação e esclarecer eventuais dúvidas.

6.3 Frequência de Contato com o Cliente

- **Validações Principais**:
  - Previstas para ocorrer apenas ao final das sprints, momento destinado à validação do progresso do projeto.
  - Como o stakeholder possui conhecimento em tecnologia, optou por não realizar reuniões semanais, considerando mais eficiente acompanhar o desenvolvimento apenas nos fechamentos de sprint.
- **Comunicação pelo WhatsApp**:
  - Em situações de dúvidas rápidas ou discussões informais ao longo do projeto, será utilizada essa ferramenta pela sua praticidade e acessibilidade.

7 Requisitos De Software

7.1 Lista de Requisitos Funcionais

ID	Nome	Descrição	Objetivo
RF01	Processar Imagens de Gabaritos	O sistema deve poder processar imagens de gabaritos preenchidos por alunos.	Automatizar a correção de gabaritos

RF02	Identificar Respostas Marcadas	O sistema deve identificar automaticamente as respostas marcadas pelos alunos nas imagens dos gabaritos.	Automatizar a correção de gabaritos
RF03	Identificar Matrícula do Aluno	O sistema deve identificar automaticamente a matrícula do aluno presente na imagem do gabarito.	Automatizar a correção de gabaritos
RF04	Enviar Gabarito Correto	O USUÁRIO deve poder enviar (upload) um arquivo .csv contendo o gabarito correto para cada questão da prova.	Automatizar a correção de gabaritos
RF05	Comparar Respostas com Gabarito	O sistema deve comparar as respostas identificadas no gabarito do aluno com o gabarito correto fornecido.	Automatizar a correção de gabaritos
RF06	Gerar Relatórios de Desempenho	O sistema deve gerar relatórios detalhados de desempenho, incluindo notas individuais, acertos, erros e estatísticas por questão.	Prover informações de desempenho
RF07	Comparar Desempenho entre Alunos	O sistema deve permitir a comparação de desempenho entre alunos individuais.	Prover informações de desempenho
RF08	Comparar Desempenho entre Grupos	O sistema deve permitir a comparação de desempenho entre grupos de alunos (ex: turmas).	Prover informações de desempenho
RF09	Salvar Relatórios na Conta	O sistema deve salvar os relatórios gerados na conta do USUÁRIO.	Prover informações de desempenho
RF10	Gerar Relatório Individual Aluno	O sistema deve gerar um relatório individual por aluno, formatado para impressão.	Prover informações de desempenho
RF11	Exportar Relatório Individual	O USUÁRIO deve poder exportar o relatório individual do aluno em formato .pdf.	Prover informações de desempenho
RF12	Criar Templates de Gabaritos	O USUÁRIO deve poder criar templates de gabaritos.	Automatizar a correção de gabaritos
RF13	Personalizar Templates	O USUÁRIO deve poder personalizar os templates de gabaritos conforme suas necessidades.	Automatizar a correção de gabaritos
RF14	Exportar Dados (CSV)	O USUÁRIO deve poder exportar os dados de correção e os relatórios gerados em formato .csv.	Prover informações de desempenho
RF15	Exportar Relatórios (PDF)	O USUÁRIO deve poder exportar os relatórios gerados em formato .pdf.	Prover informações de desempenho
RF16	Login com Google	O sistema deve permitir que o USUÁRIO realize cadastro e login utilizando sua conta Google.	Automatizar a correção de gabaritos

RF17	Associar Usuário/Escola	O sistema deve associar o nome do USUÁRIO e sua escola à conta cadastrada.	Automatizar a correção de gabaritos
RF18	Processar Pagamentos	O sistema deve integrar-se com o Stripe para processar pagamentos via Link de Pagamento.	Automatizar a correção de gabaritos
RF19	Associar Pagamento à Conta	O sistema deve associar um pagamento bem-sucedido à conta do USUÁRIO correspondente.	Automatizar a correção de gabaritos
RF20	Gerenciar Sistema de Créditos	O sistema deve gerenciar um sistema de créditos por USUÁRIO para utilização dos serviços.	Automatizar a correção de gabaritos
RF21	Adicionar Créditos à Conta	O sistema deve adicionar créditos à conta do USUÁRIO após confirmação de pagamento.	Automatizar a correção de gabaritos
RF22	Deduzir Créditos da Conta	O sistema deve deduzir créditos da conta do USUÁRIO conforme a utilização dos serviços.	Automatizar a correção de gabaritos
RF23	Visualizar Saldo de Créditos	O USUÁRIO deve poder visualizar seu saldo de créditos atual na plataforma.	Automatizar a correção de gabaritos

7.2 Lista de Requisitos Não Funcionais

ID	**Nome**	**Descrição**	**Categoria**
RNF01	Multiplataforma Desktop	O sistema deve funcionar como uma aplicação desktop nas plataformas Windows, Linux e MacOS.	Portabilidade
RNF02	Frontend Flutter	A interface do usuário (frontend desktop) deve ser desenvolvida utilizando Flutter.	Implementação
RNF03	Processamento OpenCV	O processamento de imagens dos gabaritos deve ser implementado utilizando Python e OpenCV.	Implementação
RNF04	Backend Typescript	O backend da aplicação deve ser desenvolvido utilizando Typescript e Express.js.	Implementação
RNF05	Banco Supabase	O sistema deve utilizar Supabase como plataforma de backend, com PostgreSQL.	Implementação
RNF06	Hospedagem Azure	A infraestrutura de backend deve ser hospedada em VM Linux na Azure.	Implementação
RNF07	Performance	O sistema deve processar e analisar, no mínimo, 100 gabaritos por minuto.	Desempenho
RNF08	Robustez	O sistema deve ser robusto, operando sem scripts manuais e com auto-recuperação.	Confiabilidade
RNF09	Usabilidade	A interface deve ser intuitiva e fácil de usar para diferentes perfis.	Usabilidade

RNF10	Custo Operacional	O custo operacional deve ser inferior a R$ 1,00 por aluno processado.	Operacional
RNF11	Escalabilidade	A arquitetura deve permitir aumento no volume de processamento sem reestruturação.	Escalabilidade
RNF12	Evolução Ágil	O sistema deve permitir adição ágil de novas funcionalidades com baixo acoplamento.	Manutenibilidade
RNF13	Segurança	O sistema deve garantir a segurança e confidencialidade dos dados dos alunos.	Segurança

7.3 Rastreabilidade de Requisitos

A rastreabilidade dos requisitos é mantida através das seguintes relações:

- **Requisitos → Épicos:** Cada requisito funcional está associado a um épico específico no backlog do produto.
- **Requisitos → User Stories:** Os requisitos funcionais são detalhados em histórias de usuário específicas.
- **Requisitos → Objetivos:** Cada requisito está vinculado a um objetivo específico do produto.
- **RNFs → Arquitetura:** Os requisitos não funcionais direcionam decisões arquiteturais.
- **RNFs → Tecnologias:** A escolha das tecnologias (Flutter, OpenCV, etc.) é justificada pelos RNFs.

8 DoR e DoD

O processo de validação das funcionalidades do sistema Corigge será dividido em duas etapas principais:

1. **Definition of Ready (DoR)**
    - Antes de iniciar o desenvolvimento de qualquer funcionalidade, será verificado se:
        - Um item de backlog está definido claramente. Isso implica que a User Story está escrita no formato padrão, a descrição do item é clara, concisa e compreendida por toda a equipe, e o objetivo e o valor de negócio do item são claros.
        - Os critérios de aceitação estão bem definidos. Eles devem ser testáveis, inequívocos e especificar com precisão como a funcionalidade será validada, abrangendo considerações sobre requisitos funcionais (RFs) e não funcionais (RNFs) relevantes para o item.
        - As dependências foram identificadas. Quaisquer dependências, sejam outros itens de backlog, informações externas ou decisões pendentes, precisam ser mapeadas e, idealmente, resolvidas ou ter um plano de ação estabelecido para sua resolução antes do início do desenvolvimento.

- O item possui uma estimativa de esforço. Após discussão pela equipe de desenvolvimento, o item deve ter recebido uma estimativa (por exemplo, em story points ou horas ideais), e a equipe deve concordar que o item é suficientemente pequeno para ser concluído dentro de uma única Sprint.
- A equipe possui o conhecimento técnico necessário. É preciso que a equipe possua, ou tenha planejado como adquirir, o conhecimento técnico indispensável para implementar o item, incluindo familiaridade com tecnologias como Flutter, OpenCV e Supabase.
- Se o item está alinhado com o MVP. Caso o item faça parte do escopo inicial, ele deve estar alinhado com a Definição do MVP do projeto, estabelecida na fase de Concepção.
- Por fim, a priorização do item deve estar definida. O item deve ter sido devidamente priorizado pelo Product Owner (neste contexto, o Gerente de Projeto) em relação aos demais itens presentes no backlog do projeto.

2. **Definition of Done (DoD)**

- A funcionalidade só será considerada concluída se:
  - A funcionalidade deve ter sido verificada. A funcionalidade deve estar operando conforme especificado nos critérios de aceitação em um ambiente de testes ou staging, e os requisitos não funcionais relevantes (RNF01 - Multiplataforma, RNF08 - Robustez, RNF13 - Segurança) devem ter sido atendidos dentro do escopo do item.
  - Um item de backlog é considerado "Feito"quando o código está implementado. Isso significa que todo o código necessário para a funcionalidade foi escrito, está em conformidade com os padrões de codificação da equipe e a implementação segue a arquitetura definida, como especificado na "Definição da Arquitetura"na fase de Elaboração (página 8).
  - É imprescindível que testes tenham sido realizados e aprovados. Testes unitários devem ter sido escritos e estar passando; testes de integração (por exemplo, frontend com backend, backend com OpenCV para RF03) devem ter sido escritos e estar passando; e testes de funcionalidade, conforme a fase de Construção, devem ter sido realizados e aprovados pela equipe, cobrindo todos os critérios de aceitação. Testes de performance (como o RNF07 - 100 gabaritos/min) e de usabilidade (RNF09) devem ser executados e aprovados, se aplicáveis.
  - Uma revisão de código (Peer Review) deve ter sido conduzida. O código deve ter sido revisado por pelo menos um outro membro da equipe de desenvolvimento, e todo o feedback resultante dessa revisão deve ter sido devidamente abordado e incorporado.
  - A integração contínua deve ter sido bem-sucedida. O código precisa ter sido integrado à branch principal ou de desenvolvimento do repositório, e o build de integração contínua, caso esteja configurado, deve estar passando sem erros.
  - A documentação deve estar atualizada. Qualquer documentação técnica relevante, como comentários no código, diagramas ou documentação de API, deve ter sido criada ou atualizada.

– A validação pelo Product Owner ou Cliente é crucial. A funcionalidade deve ter sido demonstrada ao Product Owner (Gerente de Projeto) e/ou ao cliente (Guia do PAS), e o Product Owner ou Cliente deve ter aceitado formalmente a funcionalidade como "Feita", conforme o "Teste de Aceitação pelo Cliente"(Seção 5.4.3) e a Validação com o Cliente, descrita na seção 5.2 do documento de Visão do Produto.
– O item deve estar livre de débitos técnicos críticos. Nenhum débito técnico crítico ou bug de alta prioridade pode ter sido introduzido ou deixado sem tratamento adequado ao finalizar o item.
– Considerações específicas do projeto Corigge devem ser atendidas. Para funcionalidades de processamento de imagem (RF01, RF02, RF03), a precisão da identificação deve atender aos níveis esperados (referência: ¿99

9 Backlog do Produto

O backlog do produto foi elaborado com base nos requisitos funcionais identificados e está organizado em épicos e histórias de usuário. A priorização foi realizada utilizando a técnica MoSCoW.

9.1 Épicos

1. **Processamento de Gabaritos (EP01)**
   - Foco no processamento e correção automática de gabaritos
   - Inclui identificação de respostas, validação e templates
2. **Geração de Relatórios (EP02)**
   - Foco na análise e geração de relatórios de desempenho
   - Inclui relatórios individuais, comparativos e estatísticas
3. **Gestão de Usuários e Pagamentos (EP03)**
   - Foco na gestão de contas, autenticação e sistema de créditos
   - Inclui integração com Stripe e controle de acesso
4. **Templates de Gabaritos (EP04)**
   - Foco na personalização e gestão de templates
   - Inclui criação, edição e definição de campos
5. **Integração e Exportação (EP05)**
   - Foco na exportação de dados e integração com outros sistemas
   - Inclui backup e diferentes formatos de exportação

9.2 Histórias de Usuário e Priorização

ID	História de Usuário	Prioridade	Horas	Épico
US01	Processar imagens de gabaritos preenchidos por alunos	Must have	40h	EP01
US02	Enviar gabarito correto e comparar com respostas	Must have	8h	EP01
US03	Validar processamento dos gabaritos	Must have	4h	EP01
US04	Identificar matrícula do aluno no gabarito	Must have	10h	EP01
US05	Gerar relatórios individuais dos alunos	Must have	8h	EP02
US11	Fazer login com conta Google e gerenciar perfil	Must have	4h	EP03
US14	Associar escola à conta	Must have	2h	EP03
US17	Criar e personalizar templates de gabaritos	Must have	12h	EP04
US18	Gerenciar templates de gabaritos	Must have	6h	EP04
US20	Definir número de questões e campos	Must have	4h	EP04
US06	Comparar desempenho entre alunos e grupos	Should have	12h	EP02
US12	Gerenciar sistema de créditos	Should have	6h	EP03
US15	Realizar pagamentos via Stripe	Should have	10h	EP03
US16	Visualizar saldo de créditos	Should have	4h	EP03
US19	Personalizar layout dos gabaritos	Should have	12h	EP04
US21	Exportar dados em diferentes formatos	Should have	4h	EP05
US22	Fazer backup dos dados	Should have	8h	EP05
US07	Visualizar estatísticas por questão	Could have	16h	EP02
US08	Acompanhar evolução do desempenho	Could have	12h	EP02
US10	Exportar relatórios gerais em PDF	Could have	4h	EP02
US13	Gerenciar diferentes níveis de acesso	Could have	12h	EP03

9.3 MVP

O Produto Mínimo Viável (MVP) inclui as seguintes funcionalidades essenciais:

- **Processamento de Gabaritos:**
  - Processar Imagens de Gabaritos (US01)
  - Correção Automática (US02)
  - Validação de Processamento (US03)
  - Identificação de Matrícula (US04)

- **Relatórios Básicos:**
  - Relatórios Individuais (US05)
- **Gestão de Usuários:**
  - Autenticação e Perfil (US11)
  - Associação Escola/Conta (US14)
- **Templates:**
  - Criação de Templates (US17)
  - Gerenciamento de Templates (US18)
  - Definição de Campos (US20)

9.4 Requisitos Não Funcionais do MVP

- Frontend: Flutter (Desktop)
- Backend: TypeScript/Express.js
- Processamento de Imagens: Python/OpenCV
- Banco de Dados: Supabase/PostgreSQL
- Hospedagem: Azure VM Linux
- Processamento mínimo de 100 gabaritos por minuto
- Interface intuitiva e fácil de usar
- Segurança e confidencialidade dos dados
- Custo operacional inferior a R$ 1,00 por aluno processado

10 Lições Aprendidas

10.1 Unidade 1

10.1.1 Organização dos Requisitos

Desafio: Desenvolver uma estrutura clara e hierárquica dos requisitos do sistema, garantindo uma priorização eficiente que atenda às necessidades do cliente e aos objetivos do projeto.

Ação de melhoria: Implementar uma ferramenta especializada em gerenciamento de requisitos para facilitar o rastreamento, versionamento e priorização das funcionalidades.

10.1.2 Escolha do Processo de Engenharia de Software

Desafio: Selecionar uma metodologia de desenvolvimento que equilibre a necessidade de flexibilidade para adaptações com a estrutura necessária para manter o projeto organizado, considerando as restrições de disponibilidade do cliente.

Ação de melhoria: Realizar uma análise prévia das necessidades do projeto, do perfil do cliente e da maturidade da equipe para escolher a metodologia mais adequada, aproveitando as experiências anteriores dos membros.

10.1.3 Organização do Tempo da Equipe

Desafio: Gerenciar eficientemente a disponibilidade dos membros da equipe para garantir a realização das reuniões de alinhamento e o cumprimento dos prazos de entrega, considerando as diferentes agendas e responsabilidades individuais.

Ação de melhoria: Implementar um sistema de gestão de tempo baseado em calendário compartilhado com horários fixos para reuniões, estabelecer marcos claros para entregas e adotar ferramentas de gestão de tarefas para monitoramento assíncrono do progresso.

10.2 Unidade 2

10.2.1 Comunicação e Organização da Equipe

Desafio: Garantir uma comunicação eficiente entre os membros da equipe, evitando mal-entendidos, retrabalhos e atrasos nas entregas, especialmente em atividades colaborativas e interdependentes. **Ação de melhoria:** Estabelecer canais de comunicação claros e centralizados (como Slack ou Discord), definir papéis e responsabilidades desde o início e promover reuniões frequentes de alinhamento com pauta e tempo definidos.

10.2.2 Uso Eficaz do GitHub e do Backlog para Gerenciamento do Projeto

Desafio: Manter o controle das tarefas, priorizações e histórico de desenvolvimento de forma organizada, garantindo rastreabilidade e visibilidade do progresso para todos os envolvidos. **Ação de melhoria:** Padronizar o uso de issues, branches e pull requests no GitHub; manter o backlog atualizado com critérios de prioridade bem definidos; e adotar quadros visuais (como Kanban no GitHub Projects ou Trello) para facilitar a visualização do andamento.

10.2.3 Integração entre Teoria e Prática

Desafio: Aplicar corretamente os conceitos aprendidos em sala de aula na execução prática do projeto, enfrentando limitações reais de tempo, recursos e conhecimento técnico. **Ação de melhoria:** Realizar revisões periódicas dos conceitos teóricos conforme a etapa do projeto, promover momentos de troca de conhecimento entre os membros da equipe e documentar o processo para fortalecer o aprendizado contínuo.

10.3 Unidade 2

10.4 Unidade 3

10.5 Unidade 4

11 Referências Bibliográficas

1. SILVA, Luciane. OpenUp: um processo integrado e ágil. Disponível em: https://medium.com/@LucianeS/openup-um-processo-integrado-e-agil-a4400c17ce62. Acesso em: 11 nov. 2024.

2. CATOLICA. METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO: OPENUP, FDD, DSDM E LEAN. Disponível em: https://conteudo.catolica.edu.br/conteudos/unileste_cursos/disciplinas/nucleo_formacao_geral/Gestao_de_projetos_e_metodos_ageis/tema_03/index.html. Acesso em: 11 nov. 2024.

3. EDUCATIVE.IO. What is a unified process model. Disponível em: https://www.educative.io/answers/what-is-a-unified-process-model. Acesso em: 11 nov. 2024.

4. EDEKI, Charles. Agile Unified Process. Disponível em: https://interhad.nied.unicamp.br/courses/roberto-pereira/ci163-projeto-de-software-ufpr-1/agenda/auppaper.pdf. Acesso em: 11 nov. 2024.

5. TREINAWEB. O que é RUP (Rational Unified Process). Disponível em: https://www.treinaweb.com.br/blog/o-que-e-rup-rational-unified-process. Acesso em: 11 nov. 2024.

6. LEFFINGWELL, D.; WIDRIG, D. Managing Software Requirements: A Use Case Approach. 2. ed. Addison-Wesley, 2003.

7. AMBLER, S. Agile Modeling. Wiley, 2002.

8. LEFFINGWELL, Dean. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. 1. ed. Addison-Wesley Professional, 2011.

9. PATTON, Jeff; ECONOMY, Peter. User story mapping: discover the whole story, build the right product. O'Reilly Media, Inc., 2014.

10. SCHWABER, Ken; SUTHERLAND, Jeff. O Guia do Scrum: O Guia Definitivo para o Scrum: As Regras do Jogo. Novembro de 2020.