Meu padrinho universitário

Cliente: *Unifei*

Revisões do Documento

Revisões são melhoramentos na estrutura do documento e também no seu conteúdo. O objetivo primário desta tabela é a fácil identificação da versão do documento. Toda modificação no documento deve constar nesta tabela.

Data	Versão	Descrição	Autor
26/08/2025	0.1	Versão base com requisitos para análise	Time
28/10/2025	0.2	Versão atualizada com ajuste de requisitos e padronização	Time
01/11/2025	0.3	Versão atualizada com base nas sugestões do professor	Time

Auditorias do Documento

Auditorias são inspeções conduzidas o SEPG – Software Engineer Process Group (Grupo de Engenharia de Processo de Software), e tem por objetivo garantir uma qualidade mínima dos artefatos gerados durante o processo de desenvolvimento. Essa tabela pode ser utilizada também pelo GN – Gerente da Área de Negócio com o objetivo de documentar a viabilidade do mesmo.

Data	**Versão**	**Descrição**	**Autor**
08/09/2025	0.1	Avaliação do grupo 10 a respeito do DRE	Grupo 10

Introdução

Este documento especifica os requisitos do software Meu Padrinho Universitário, fornecendo aos desenvolvedores as informações necessárias para a execução de seu projeto e implementação, assim como para a realização dos testes e homologação.

Esta introdução fornece as informações necessárias para fazer um bom uso deste documento, explicitando seus objetivos e as convenções que foram adotadas no texto.

a) Convenções, termos e abreviações

A correta interpretação deste documento exige o conhecimento de algumas convenções e termos específicos, que são descritos a seguir.

- **VETERANO**: Estudante com experiência na universidade.
- **CALOURO**: Estudante recém-ingressante na universidade.
- **MATCH**: Processo de vinculação entre veterano e calouro, realizado pelo sistema.
- **Período de maturação:** período em que o sistema ainda não conectou os veteranos aos calouros e novos cadastros podem ser feitos.
- **PADUNI**: Sigla de "Padrinho Universitário", nome do sistema.
- **MFA:** autenticação multifator.

b) Identificação dos Requisitos

Por convenção, a referência a requisitos é feita através do identificador do requisito, de acordo com o esquema abaixo:

[identificador de tipo de requisito.identificador do requisito]

O identificador de tipo de requisito pode ser:

- RF: Requisito funcional
- RNF: Requisito não-funcional

Identificador do requisito é um número, criado sequencialmente, que determina que aquele requisito é único para um determinado tipo de requisito.

Ex: RF001, RF002, RNF001, RNF002.

c) Identificação dos Requisitos

Para estabelecer a prioridade dos requisitos foram adotadas as denominações “essencial”, “importante” e “desejável”.

- **Essencial** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
- **Importante** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- **Desejável** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis são requisitos que podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

Visão Geral do Produto

Este documento especifica os requisitos do software **Meu Padrinho Universitário (PADUNI)**, fornecendo aos desenvolvedores as informações necessárias para sua execução, implementação, testes e homologação.

O sistema busca unir veteranos e ingressantes (calouros) da universidade, promovendo integração, troca de experiências e acolhimento no ambiente acadêmico.

a) Abrangência e sistemas relacionados

O PADUNI é um aplicativo web que deverá possuir relação com uma API de IA generativa, que terá como função realizar a associação entre veteranos e calouros, e serviços de email para envio de notificações automáticas (SMTP).

Principais funcionalidades:

- *Cadastro de usuários (veteranos e calouros).*
- *Processo de "match" automático entre veterano e calouro feito através de IA.*
- *Envio de emails notificando o match.*
- *Chat interno para comunicação entre padrinho (veterano) e afilhado (calouro).*

b) Descrição do cliente

*O cliente é a **Universidade Federal de Itajubá (Unifei)**, que busca soluções para melhorar a integração entre alunos ingressantes e veteranos. Todo ano há o processo de apadrinhamento, realizado geralmente através de um google forms, onde são associados, manualmente, padrinhos (veteranos) a seus apadrinhados (calouros). O processo não utiliza como base os interesses de ambos os usuários, e por esse motivo, a taxa de adesão ao processo costuma ser baixa e após o processo, grande parte dos associados não mantém o contato durante longos períodos. O apadrinhamento tem como objetivo facilitar a vida do calouro, já que quaisquer dúvidas sobre processos, localizações de sala, empresas juniores, equipes de competição, tudo relacionado à universidade, pode ser sanada com a ajuda de seu padrinho, diminuindo a barreira de entrada de universidades federais como a Unifei*

c) Descrição dos usuários

Administradores: alunos associados ao DACOMP, ou qualquer atlética de cursos situados na Unifei.
***Veteranos**: alunos com 1 ou mais anos de experiência, responsáveis por orientar calouros.*
***Calouros**: alunos ingressantes que buscam apoio e integração na universidade.*

Requisitos Funcionais

[RFC01] Manter Usuário

Ator: Sistema

O sistema deve permitir a criação, visualização, edição e deleção de novos usuários sejam eles padrinhos ou apadrinhados de acordo com as regras abaixo:

Regras:

1. A criação deve ser feita pelo próprio usuário através de um formulário de inscrição disponível na tela “Cadastre-se”.

2. A edição das informações também só pode ser feita pelo próprio usuário.

3. A edição de senha e email deve ser realizada com a etapa adicional de MFA, utilizando o email cadastrado para o envio de um código que precisará ser validado.

4. O usuário deve ser impedido de visualizar as informações de outro usuário, a não ser que ele seja o padrinho/apadrinhado do mesmo.

5. A deleção pode ser feita pelo próprio usuário ou por um administrador, no segundo caso, deve ser fornecido um motivo para a ação, que será armazenado para futura auditoria caso necessário.

[RFS01] Cadastrar Usuário

Atores: Calouros, Veteranos
Qualquer estudante da Universidade Federal de Itajubá, tanto veteranos quanto calouros, poderá se cadastrar no sistema caso tenha interesse em ser padrinho ou apadrinhado. Para isso serão exigidos os seguintes campos no ato da inscrição, na tela de cadastro:

Campo	Descrição	Regras
Nome *	Nome do aluno	Campo simples de texto limite de 30 caracteres
Email *	Email do aluno	Verificação de sintaxe de email válido. Será único por aluno
Ano de nascimento *	Utilizado para cálculo da idade do aluno	Campo simples, número inteiro
Gênero *	Gênero do aluno	Campo select, com opções masculino, feminino e outro
Curso *	Curso do aluno	Select de cursos já existentes

Ano de ingresso na Unifei *	Ano de ingresso, o sistema vai definir automaticamente se é calouro ou veterano	Regra para cálculo de aluno com mínimo de 16 anos, abaixo da idade não deve permitir criação
Interesses*	interesses do aluno	Campo de texto com limite de 600 caracteres, digitação livre. Será utilizado pela IA para análise de opção de match
Senha *	Senha que será utilizada para fazer login	Deve ser forte: ● Mais de 8 caracteres ● Ao menos 1 maiúscula ● Ao menos 1 minúscula ● Ao menos 1 símbolo
Confirmação de senha *	Confirmação da senha digitada anteriormente	Deve ser igual a senha digitada anteriormente

Tabela 1: campos a serem preenchidos no cadastro

Campos com * são campos obrigatórios. O sistema irá criticar e impedir que a entidade seja cadastrada caso o campo esteja vazio. Todos os campos que requerem validação também deverão estar preenchidos adequadamente. Caso contrário, a entidade também não será cadastrada.

Processamento:

1. Verificar preenchimento dos campos obrigatórios.
2. Validar regras de conteúdo (formato do email, força da senha, idade mínima de 16 anos).
3. Determinar se o usuário é calouro ou veterano com base no ano de ingresso.
4. Verificar a unicidade do email.
5. Criar registro no banco de dados.

Saídas:

- Mensagem de sucesso.
- Usuário cadastrado no sistema.
- Redirecionamento à tela inicial.

Requisitos Técnicos:

- Banco de dados relacional (PostgreSQL ou similar).

- Validação de formulário no frontend e backend.
- Utilização de Bcrypt para hash de senha.
- API REST para criação de usuário.

Critérios de Aceitação:

- Nenhum campo obrigatório pode ser nulo.
- Email duplicado deve ser recusado.
- A senha deve ser armazenada como hash no banco.
- O cadastro deve ocorrer em até 2 segundos.

Dependências:

- Nenhuma.

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFS02] Fazer login

Atores: Calouros, Veteranos
O sistema deverá permitir que o usuário acesse o sistema após já estar cadastrado no mesmo. As credenciais de acesso devem ser o email e a senha cadastrados. O backend será responsável por validar ambos os campos, incluindo a senha encriptada, e no caso de erro, retornar a seguinte mensagem ao usuário: “Credenciais inválidas, verifique as informações”. No caso de sucesso, o usuário terá acesso à página inicial da aplicação. Caso o usuário tente fazer login sucessivas vezes com falha, após 10 tentativas o acesso será bloqueado por 30 minutos, a fim de evitar ataques de sobrecarregamento da API.

Entradas:

- Email
- Senha

Processamento:

- Receber as informações via endpoint
- Validar email e senha no banco de dados
- Devolver resposta para o frontend

Saídas:

- Confirmação de atualização.
- Caso sucesso: redirecionar o usuário para a tela principal.
- Caso fracasso: informar o usuário sobre a falha sem mencionar qual campo está incorreto.

Requisitos Técnicos:

- Autenticação JWT.
- Criptorafia de senha com bcrypt.

Critérios de Aceitação:

- Usuário não deve ser informado de qual campo está com falha.
- O token de acesso deve ser JWT, enviado ao frontend na resposta da requisição.
- Tentativas de login sucessivas (10 tentativas em menos de 2 minutos) devem ser consideradas ataques, e travar o sistema por 30 minutos.

Dependências:

- Nenhuma

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFS03] Visualizar Perfil de Usuário

Atores: Calouros, Veteranos

O sistema deverá permitir que o usuário visualize as suas informações cadastradas no sistema. Todos os dados serão exibidos em modo "somente leitura", não sendo possível a edição nesta funcionalidade. O usuário deve estar logado para acessar esta tela. Todos os campos da tabela 1, disponível na seção RFS01, serão exibidos, menos a senha e a confirmação de senha.

Entradas (para Visualização):

- Nenhuma (funcionalidade acessada após login, usuário identificado via Token JWT).

Processamento:

1. Validar se o usuário está autenticado (verificar a validade do token JWT).
2. Identificar o usuário (ex: pelo ID contido no token).
3. Buscar no banco de dados as informações cadastrais associadas a esse usuário.
4. Retornar os dados para o frontend para exibição.

Saídas:

- Tela de perfil preenchida com os dados do usuário (listados na tabela acima).
- Caso o usuário não esteja autenticado, deve ser redirecionado para a tela de Login [RFS02].

Requisitos Técnicos:

- Autenticação JWT (o endpoint deve ser protegido).
- API REST (endpoint GET para buscar os dados do perfil).
- Banco de dados relacional (para consulta dos dados).

Critérios de Aceitação:

- O usuário só pode visualizar o seu próprio perfil.
- As informações exibidas devem ser idênticas às registradas no banco de dados.
- Nenhum campo deve permitir a edição nesta tela.
- O acesso à tela de perfil só deve ser permitido para usuários autenticados.

Dependências:

- Nenhuma

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFS04] Editar Perfil de Usuário

Atores: Calouros, Veteranos

O sistema deverá permitir que o usuário, após autenticado, atualize suas informações cadastrais. Os campos editáveis são: Nome, Ano de nascimento, Gênero, Curso, Ano de ingresso e interesses. Para confirmar e salvar qualquer alteração, será obrigatório que o usuário insira sua "Senha Atual" para validação. Campos não listados, como o Email, não poderão ser alterados por esta funcionalidade.

Entradas:
O usuário poderá fornecer novos valores para os campos da tabela 2, que segue abaixo. A "Senha Atual" é sempre obrigatória para salvar as alterações solicitadas.

Campo	Regras
Nome	Campo simples de texto limite de 30 caracteres
Ano de nascimento	Campo simples, número inteiro
Gênero	Campo select, com opções masculino, feminino e outro
Curso	Texto livre (máx. 50 caracteres). Após 5 caracteres, busca cursos existentes (LIKE, sem case sensitive) e sugere via autocomplete. Usuário pode selecionar um curso sugerido ou digitar um novo.

	Se já existir, associa ao existente; senão, cadastra novo.
Ano de ingresso na Unifei	Regra para cálculo de aluno com mínimo de 16 anos, abaixo da idade não deve permitir criação
interesses	Campo de texto com limite de 600 caracteres, digitação livre. Será utilizado pela IA para análise de opção de match
Senha atual *	Senha será utilizada para confirmar as alterações

Tabela 2: campos sujeitos a atualização pelo usuário.

Processamento:

1. Validar se o usuário está autenticado (via token JWT).
2. Receber os novos dados preenchidos e a "Senha Atual".
3. Validar a "Senha Atual" comparando-a com o hash armazenado no banco (usando bcrypt).
4. Caso a "Senha Atual" esteja incorreta, retornar mensagem de erro (ex: "Senha inválida.") e interromper o processo.
5. Caso a "Senha Atual" esteja correta, validar os campos alterados (Nome, Ano de nascimento, etc.) conforme as regras de cadastro [RFS01].
6. Se houver falha na validação de algum campo, retornar erro e não salvar. Informar ao usuário qual campo está com problema.
7. Se todas as validações forem aprovadas, atualizar os dados no registro do usuário no banco de dados.

Saídas:

- Mensagem de "Perfil atualizado com sucesso."
- Mensagem de falha, caso a senha esteja incorreta ou algum campo viole as regras de negócio.

Requisitos Técnicos:

- Autenticação JWT (o endpoint deve ser protegido).
- Utilização de Bcrypt para validação da senha atual.
- API REST para atualização de perfil (endpoint PUT ou PATCH).
- Validação de formulário no frontend e backend.

Critérios de Aceitação:

- A alteração deve ser confirmada apenas com a "Senha Atual" válida.
- A atualização deve refletir no banco de dados imediatamente.
- O usuário só pode editar o seu próprio perfil.
- O email do usuário não pode ser alterado por esta funcionalidade.

- A senha do usuário não pode ser alterada por esta funcionalidade

Dependências:

- [RFS01] Cadastrar Usuário
- [RFS02] Fazer login

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFS05] Alterar email ou senha

Atores: Calouros, Veteranos

O sistema deverá permitir que o usuário autenticado altere seu email de cadastro ou sua senha de acesso. Ambas as operações são consideradas críticas e exigirão um passo de verificação em duas etapas (MFA) via email para garantir a segurança.

Entradas:

O usuário deverá preencher os campos obrigatórios e pelo menos um dos campos opcionais (Novo Email ou Nova Senha) da tabela 3, disponível abaixo.

Campo	Descrição	Regras
Novo Email	Novo email do aluno (opcional)	Se preenchido, deve ter sintaxe de email válida. Deve ser único no sistema (verificar duplicidade).
Nova Senha	Nova senha de acesso (opcional)	Se preenchida, deve ser forte (regras do RFS01: >8 caracteres, 1 maiúscula, 1 minúscula, 1 símbolo).
Confirmação de Nova Senha	Confirmação da nova senha digitada	Se 'Nova Senha' foi preenchida, deve ser idêntica.
Senha Atual *	Senha atual usada para confirmar a identidade	Obrigatória. Deve ser válida (comparada com o hash do banco).

Código MFA *	Código de verificação enviado ao email	Obrigatório. Deve ser o código ativo enviado por email.

Tabela 3: campos para alteração de email ou senha do usuário.

Processamento:

1. **Etapa 1: Solicitação de Código**
    - Usuário autenticado solicita o código MFA através de um botão “Enviar código para email”.
    - Sistema gera um código único (ex: 6 dígitos) e armazena-o associado ao usuário com um timestamp de expiração (5 minutos).
    - Sistema envia o código para o email *atualmente cadastrado* do usuário.
    - Sistema informa o usuário que o código foi enviado.
2. **Etapa 2: Confirmação da Alteração**
    - Receber os dados do formulário via endpoint.
    - Validar a "Senha Atual" (comparando com hash do banco via bcrypt). Se falhar, rejeitar.
    - Validar o "Código MFA": verificar se corresponde ao código armazenado e se não expirou (5 minutos). Se falhar, rejeitar.
    - Validar campos preenchidos:
        - Se "Novo Email": verificar formato e unicidade no banco.
        - Se "Nova Senha": verificar regras de força e se a confirmação é idêntica.
    - Se todas as validações passarem:
        - Atualizar o email e/ou o hash da nova senha (bcrypt) no banco de dados.
        - Invalidar o código MFA (marcar como usado).
        - Invalidar todas as sessões ativas (tokens JWT) do usuário.

Saídas:

- **Etapa 1:** Mensagem de "Código de verificação enviado para o seu email."
- **Etapa 2 (Sucesso):** Mensagem de "Dados atualizados com sucesso. Por favor, faça login novamente." Redirecionamento para a tela de login.
- **Etapa 2 (Falha):** Mensagem de erro específica (ex: "Senha atual inválida.", "Código de verificação incorreto ou expirado.", "Esse email já está em uso.").

Requisitos Técnicos:

- Autenticação JWT (endpoint protegido).
- Sistema de autenticação multifator (MFA) via email (serviço de envio de email, ex: SMTP).
- Armazenamento temporário (cache ou banco) para o código MFA e sua expiração.

- Criptografia de senhas (bcrypt) para validar a senha atual e salvar a nova.
- Lógica de invalidação de tokens JWT (ex: blacklist de tokens).

Critérios de Aceitação:

- O código MFA deve expirar após 5 minutos.
- O código MFA só pode ser utilizado uma vez.
- A alteração (email ou senha) só deve ser efetivada se a "Senha Atual" E o "Código MFA" estiverem corretos.
- A atualização bem-sucedida de email ou senha deve invalidar todas as sessões antigas, forçando o usuário a fazer login novamente.

Dependências:

- [RFS01] Cadastrar Usuário
- [RFS02] Fazer login

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFS06] Excluir Conta

Atores: Calouros, Veteranos

O sistema deverá permitir que o usuário autenticado solicite a exclusão permanente de sua conta. Esta é uma ação irreversível e, para garantir a segurança, exigirá uma confirmação em duas etapas: a "Senha Atual" do usuário e um "Código de Confirmação" enviado ao email cadastrado.

Uma vez confirmada, todos os dados pessoais do usuário (perfil, interesses, etc.) serão excluídos integralmente e de forma irreversível do sistema, em conformidade com a LGPD.

Entradas:

- Email
- Código de confirmação.

Processamento:

1. **Etapa 1: Solicitação de Código**
    - Usuário autenticado acessa a tela de exclusão e solicita a operação.
    - Sistema gera um código de confirmação (ex: 6 dígitos) e armazena-o com um timestamp de expiração (ex: 5 minutos).
    - Sistema envia o código para o email cadastrado do usuário, alertando que se trata de uma exclusão permanente.
2. **Etapa 2: Confirmação da Exclusão**

    - Receber a "Senha Atual" e o "Código de Confirmação" via endpoint.
    - Validar a "Senha Atual" (comparando com hash do banco via bcrypt). Se falhar, rejeitar.
    - Validar o "Código de Confirmação": verificar se corresponde ao código armazenado e se não expirou. Se falhar, rejeitar.
    - Se ambas as validações passarem:
        - Registrar a ação em log de ações críticas [RF05.03].
        - Executar a rotina de exclusão (hard delete) de todos os dados do usuário no banco de dados.
        - Invalidar a sessão (token JWT) do usuário.

Saídas:

- **Etapa 1:** Mensagem de "Código de confirmação enviado para o seu email."
- **Etapa 2 (Sucesso):** Mensagem de "Conta excluída com sucesso." Redirecionamento para a tela de login.
- **Etapa 2 (Falha):** Mensagem de erro (ex: "Senha atual inválida.", "Código de confirmação incorreto ou expirado.").

Requisitos Técnicos:

- Autenticação JWT (endpoint protegido).
- Sistema de envio de email automatizado (MFA).
- Utilização de Bcrypt (para validar "Senha Atual").
- Rotina de exclusão de dados (Hard Delete) em conformidade com a LGPD (Lei Geral de Proteção de Dados).

Critérios de Aceitação:

- A exclusão só deve ser efetivada se a "Senha Atual" E o "Código de Confirmação" estiverem corretos.
- O código de confirmação deve expirar após o tempo definido (ex: 5 minutos).
- Todos os dados pessoais do usuário devem ser removidos integralmente do banco de dados.
- Após a exclusão, o usuário não deve mais conseguir autenticar-se no sistema com as credenciais antigas.

Dependências:

- [RFS01] Cadastrar Usuário
- [RFS02] Fazer login

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFC02] Realizar Match

[RFS07] Realizar match Automático

Atores: Sistema
Realizar o processo de vinculação entre calouros e veteranos com base na compatibilidade de interesses e curso, de forma automática através de uma inteligência artificial, que deverá ser abastecida com o conteúdo dos usuários e exportar a combinação ideal entre eles.

Abaixo estão as regras que serão adicionadas no prompt para a escolha do match ideal, a ordem indica a hierarquia de quais regras são mais importantes, da mais importante para a menos. A única regra essencial será de alunos serem do mesmo curso, com o objetivo de evitar com que alunos de cursos diferentes se associem, já que o auxílio de um veterano deve ser ajudar os calouros a passar pelas mesmas experiências que ele, e em cursos diferentes a maioria das vezes isso não ocorre. Na seção RFC05 há mais detalhes técnicos de como a IA irá atuar.

Seguem abaixo a lista de regras por importância, começando da mais para a menos importante:

1. Alunos do mesmo curso (essencial);
2. Alunos de mesmo gênero;
3. Alunos de mesma idade;
4. Alunos com os mesmos interesses;

Entradas:

- Lista de Calouros (usuários com status "apadrinhado") que ainda não possuem match, contendo: nome, curso, gênero, interesses e idade.
- Lista de Veteranos (usuários com status "padrinho") disponíveis para apadrinhamento, contendo: nome, curso, gênero, interesses e idade.

Processamento:

- O sistema (através de uma rotina agendada) inicia o processo de match.
- O sistema identifica todos os calouros ativos que necessitam de um padrinho.
- O sistema identifica todos os veteranos disponíveis para serem padrinhos.
- O módulo de IA é acionado, recebendo os dados dos perfis.
- A IA calcula um "score de compatibilidade" entre os calouros e os veteranos disponíveis, com base nos critérios (interesses, curso, gênero, idade).

- O sistema gera as correspondências (pares) com os maiores scores de similaridade, priorizando a qualidade do match.
- O sistema registra os vínculos (calouro-veterano) aprovados no banco de dados.

Saídas:

- Vínculos de "padrinho-apadrinhado" registrados na tabela de relacionamentos do banco de dados.
- Atualização do status dos usuários envolvidos (ex: "Match Realizado").
- Trigger para envio de emails de notificação.

Requisitos Técnicos:

- Módulo de IA para cálculo de compatibilidade e similaridade.
- Banco de dados (ex: PostgreSQL) com os perfis de usuário.
- Rotina automatizada (ex: Job/Cron) para disparar o processo de match periodicamente.

Critérios de Aceitação:

- Cada calouro que solicitou um padrinho deve receber pelo menos uma sugestão de match válida (pressupondo que existam veteranos compatíveis disponíveis).
- A IA deve priorizar a compatibilidade de "interesses" e "Curso".

Dependências:

- RFC01 (Manter Usuário).
- RFC05 (IA).

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFS08] Notificar Match

Atores: Sistema

Após a conclusão bem-sucedida de um vínculo pelo [RFS07] (Match Automático), o sistema deverá notificar ambos os usuários (calouro e veterano) sobre o novo match. A notificação deve ser um email automático que não contenha dados pessoais (como nome, curso, etc.) do parceiro. O objetivo do email é apenas informar sobre o evento e direcionar o usuário a fazer login na aplicação para visualizar os detalhes.

Entradas (Dados para o Processamento):

- ID e Email do Calouro que recebeu o match.
- ID e Email do Veterano que recebeu o match.

Processamento:

1. O processo é iniciado (disparado) imediatamente após o [RFS07] registrar um novo par (calouro-veterano) no banco de dados.
2. O sistema obtém os endereços de email de ambos os usuários envolvidos.
3. O sistema carrega o template de email padrão para "Notificação de Novo Match".
4. O sistema envia o email para o endereço do calouro.
5. O sistema envia o email para o endereço do veterano.

Saídas:

- Emails de notificação efetivamente enviados (e presumivelmente recebidos) pelos usuários.

Requisitos Técnicos:

- Serviço de envio de email (ex: SMTP, ou APIs como SendGrid, SES).
- Template de email (HTML/Texto) pré-definido.
- Mecanismo de *trigger* (ex: listener de evento de aplicação ou *trigger* de banco) para iniciar o processo após o match.

Critérios de Aceitação:

- O email deve ser enviado para ambos os usuários em até 1 minuto após a criação do match no banco de dados.
- O conteúdo do email não deve conter nenhum dado pessoal (nome, curso, interesses, etc.) do parceiro; deve apenas incluir um link ou instrução para acessar a aplicação.

Dependências:

- RFS07 (Match Automático).

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RFS09] Desfazer Match

Atores: Administrador
O sistema deve permitir que o administrador desfaça um match manualmente em casos específicos (ex: incompatibilidade relatada por um dos lados, com justificativa), as motivações estarão melhor detalhadas no RFS10. Quando essa ação for tomada, os usuários não serão

notificados via email e também não estarão sujeitos a receber um novo match. Caso o administrador decida que a justificativa não é suficiente para desfazer o match, ele deve direcionar o aluno solicitante para que o mesmo delete sua conta.

Entradas:

- Identificação do match.
- Justificativa textual.

Processamento:

- Validar justificativa.
- Atualizar status do match.

Saídas:

- Mensagem de confirmação para o usuário na tela (ex: "Sua solicitação de anulação foi enviada ao administrador para análise.").
- Email de notificação enviado com sucesso ao Administrador.

Requisitos Técnicos:

- Autenticação JWT (a funcionalidade está em uma área logada).
- Serviço de envio de email (SMTP ou API).
- API REST (endpoint POST) para submeter a solicitação de anulação.
- Tabela no banco de dados para registrar as solicitações pendentes de revisão.

Critérios de Aceitação:

- O usuário só pode solicitar a anulação de um match que esteja atualmente ativo para ele.
- A justificativa textual é obrigatória para enviar a solicitação.
- O email para o administrador deve ser enviado imediatamente após o registro da solicitação.

Dependências:

- [RFS07] Match Automático (para que exista um match a ser anulado).

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RFS10] Solicitar anulação de match

Atores: Calouros, Veteranos

O sistema deverá permitir que o usuário (calouro ou veterano), após ter um vínculo criado pelo [RFS07], solicite a anulação desse match. A opção estará disponível na tela "Meu Match". Para solicitar, o usuário deverá obrigatoriamente fornecer uma justificativa textual.

Após o envio da solicitação, o sistema não desfaz o match automaticamente. Em vez disso, ele registra a solicitação e notifica o Administrador por email, enviando os detalhes (email do solicitante e a justificativa) para que o Administrador tome uma ação manual (conforme [RFS09]).

Entradas:

- Email do aluno.
- Justificativa textual.

Processamento:

1. O usuário autenticado acessa a tela "Meu Match", onde visualiza seu par atual.
2. O usuário seleciona a opção "Solicitar Anulação de Match".
3. O sistema exige o preenchimento do campo "Justificativa".
4. Após o envio, o sistema registra a solicitação (ex: em uma tabela "solicitacoes_anulacao") com o ID do usuário solicitante, o ID do match e o texto da justificativa.
5. O sistema dispara um email automático para o endereço de email do Administrador.
6. O email enviado ao Administrador deve conter:
    - O email do usuário solicitante.
    - O texto completo da "Justificativa".
    - (Opcional) Um link para o painel de administração (referente ao [RFS09]).

Saídas:

- Mensagem de confirmação para o usuário na tela (ex: "Sua solicitação de anulação foi enviada ao administrador para análise.").
- Email de notificação enviado com sucesso ao Administrador.

Requisitos Técnicos:

- Autenticação JWT (a funcionalidade está em uma área logada).
- Serviço de envio de email (SMTP ou API).
- API REST (endpoint POST) para submeter a solicitação de anulação.
- Tabela no banco de dados para registrar as solicitações pendentes de revisão.

Critérios de Aceitação:

- O usuário só pode solicitar a anulação de um match que esteja atualmente ativo para ele.
- A justificativa textual é obrigatória para enviar a solicitação.
- O email para o administrador deve ser enviado imediatamente após o registro da solicitação.

Dependências:

- [RFS07] Match Automático (para que exista um match a ser anulado).
- [RFS09] Desfazer Match (Administrativo).

[RFC03] Mediar comunicação entre Usuários

[RFS11] Enviar e Receber Mensagens

Atores: Calouros, Veteranos

O sistema deverá disponibilizar um chat interno (na tela "Meu Match") que permita a comunicação em tempo real exclusivamente entre o padrinho e seu afilhado (usuários com vínculo ativo pelo [RFS07]). O chat deve permitir o envio e recebimento de mensagens de texto.

Campo	Descrição	Regras
Mensagem	O conteúdo textual da mensagem.	Limite de 2000 caracteres.
Remetente	ID do usuário que está enviando.	Obtido do usuário autenticado (JWT).
Destinatário	ID do usuário que receberá (o par do match).	Obtido o vínculo do match.
Timestamp	Data e hora do envio.	Gerado automaticamente pelo servidor no momento do recebimento.

Tabela 4: campos de entrada para envio de mensagem.

Entradas:

- Quando um usuário envia uma mensagem, os seguintes dados são capturados

Processamento:

1. O usuário (Remetente) digita uma mensagem no componente de chat e envia.
2. O sistema (via WebSocket/Realtime DB) recebe a mensagem.
3. O sistema valida se o Remetente e o Destinatário possuem um match ativo. Se não, a mensagem é rejeitada.
4. O sistema armazena a mensagem (conteúdo, remetente, destinatário, timestamp) no banco de dados.
5. O sistema transmite a mensagem em tempo real para o Destinatário (se estiver online).

6. A interface do chat do Remetente e do Destinatário é atualizada para exibir a nova mensagem.

Saídas:

- Mensagem exibida na interface de chat do Remetente e do Destinatário.
- (Opcional) Notificação em tempo real (ex: "nova mensagem") se o destinatário estiver com o app aberto, mas fora do chat.

Requisitos Técnicos:

- Tecnologia de comunicação em tempo real (ex: WebSocket, Socket.IO, ou Firebase Realtime Database/Firestore).
- Banco de dados (NoSQL ou Relacional) otimizado para armazenar e consultar o histórico de mensagens.
- API REST (para buscar histórico de mensagens) e API de Realtime (para envio/recebimento).

Critérios de Aceitação:

- O atraso máximo entre o envio e o recebimento da mensagem (com ambos os usuários online) deve ser de até 1 segundo.
- Usuários só podem enviar mensagens para o seu par (padrinho ou afilhado) com match ativo.
- O histórico de mensagens deve ser carregado corretamente ao abrir o chat.

Dependências:

- [RFS07] Match Automático.

Prioridade:
[ ] Essencial [] Importante [X] Desejável

[RFC04] Gerenciar Administração do Sistema

[RFS12] Visualizar e excluir Usuários

Atores: Administrador
Permitir que o administrador visualize e exclua perfis de usuários. No caso de exclusões, o administrador deve atribuir um motivo para a exclusão a fim de auditorias.

Entradas:

- ID do usuário

- Motivo da ação.

Processamento:

- Executar ação (visualizar/excluir).
- Registrar no log.

Saídas:

- Confirmação da ação.

Requisitos Técnicos:

- Painel administrativo protegido.

Critérios de Aceitação:

- Exclusão registrada corretamente.

Dependências:

- Nenhuma

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFS13] Armazenar log de ações críticas

Atores: Sistema

O sistema deverá registrar (auditar) todas as ações críticas executadas por Administradores em uma tabela de log dedicada. Ações críticas incluem, mas não se limitam a: visualização de perfis de usuário, exclusão de contas ([RFS06]) e desfazimento de matches ([RFS09]).

Esta tabela de log será "append-only" (apenas adição), não permitindo a exclusão ou alteração de registros, nem mesmo por Administradores. Administradores poderão apenas *visualizar* esses logs através do painel administrativo. Para otimizar o armazenamento e evitar gargalos, um backup dos logs deve ser realizado a cada três meses e arquivado em um serviço de armazenamento frio (Amazon S3 Glacier).

Entradas: Quando uma ação crítica é executada, os dados devem ser capturados e persistidos

Campo	Descrição
Responsável	Administrador responsável pela ação
Ação	Qual foi a ação executada
Data e hora	Quando a ação foi executada
Alvo	Quem foi o perfil alvo da execução
Justificativa	Qual foi o motivo da ação

Tabela 5: campos que devem ser armazenados durante ações críticas.

Somente administradores poderão visualizar essa tabela, porém não poderão excluir nenhuma informação dela. Para otimizar o armazenamento e evitar gargalos, a cada três meses um backup deve ser realizado em um Amazon S3 Glacier.

Entradas:

- Responsável
- Ação
- Data/hora
- Alvo
- Justificativa.

Processamento:

1. Quando um Administrador executa uma ação crítica (ex: via [RFS06] ou [RFS09]), o endpoint dessa funcionalidade deve, além de sua lógica principal, disparar um evento de log.
2. O sistema de log recebe os dados do evento (Responsável, Ação, Alvo, Justificativa).
3. O sistema insere um novo registro (imutável) na tabela de logs.
4. Uma rotina agendada (ex: Cron Job) deve ser executada a cada 3 meses para:
    - Gerar um arquivo de backup (ex: CSV, JSON) dos logs do período.
    - Realizar o upload desse arquivo para o bucket S3 Glacier.
    - (Opcional, dependendo da política de retenção) Limpar (prune) os registros que já foram backupeados do banco de dados principal.
-

Saídas:

- Registro de log persistido de forma segura e imutável no banco de dados.
- Arquivo de backup enviado ao Amazon S3 Glacier a cada 3 meses.

Requisitos Técnicos:

- Banco de dados seguro para logs.
- Serviço de agendamento de tarefas (Cron job) para a rotina de backup.
- Integração com API do Amazon S3 Glacier (ou S3 com política de ciclo de vida para Glacier).
- (Implícito) Painel administrativo para *visualização* (somente leitura) dos logs.

Critérios de Aceitação:

- O registro de log não pode ser apagado ou alterado via aplicação após ser criado.
- O backup para o S3 Glacier deve ser concluído com sucesso a cada 3 meses.
- Todas as ações críticas (como [RFS06] e [RFS09]) devem obrigatoriamente gerar um registro de log correspondente.

Dependências:

- Nenhuma

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RFC05] Acionar Inteligência Artificial

[RFS14] Definir data de execução da IA

Atores: Administrador

O administrador deverá poder definir, através do painel administrativo, uma data e hora específicas para que a rotina de Match Automático (IA) [RFS07] seja executada. O sistema só permitirá o agendamento de *uma* execução futura por vez; se uma nova data for definida antes da anterior ocorrer, a nova data substituirá a antiga.

Entradas:

- Data e hora da execução.
- Identificação do administrador responsável.
- Timestamp da solicitação.

Processamento:

1. O administrador, autenticado no painel, acessa a funcionalidade de agendamento da IA.
2. O sistema valida se o usuário possui permissão de Administrador.
3. O Administrador insere a "Data e Hora da Execução" desejada.
4. O sistema (backend) valida o formato da data/hora e verifica se é uma data futura.
5. O sistema valida se já existe um agendamento pendente. Se sim, ele será substituído.

6. O sistema registra no banco de dados o agendamento da execução da IA (ex: em uma tabela de configuração de jobs).
7. O sistema registra a ação de agendamento no log [RFS13], identificando o Administrador responsável.
8. O sistema retorna a confirmação para a interface do Administrador.

Saídas:

- Notificação visual ao administrador (ex: “Execução da IA agendada para [data/hora].”).
- Registro do agendamento persistido no banco de dados.

Requisitos Técnicos:

- Interface administrativa segura (com autenticação JWT).
- API REST (endpoint POST/PUT) para registrar o agendamento.
- Banco de dados para armazenamento do agendamento.
- Validação de formato e regras de data/hora no backend.

Critérios de Aceitação:

- Não deve ser possível agendar uma data ou hora no passado.
- O sistema deve confirmar o agendamento na interface em até 2 segundos.
- Apenas administradores autenticados podem definir a data.
- Só pode existir um agendamento futuro por vez; um novo agendamento substitui o anterior.

Dependências:

- [RFS07] Match Automático (É a rotina de IA que será agendada).

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFS15] Modificar data de execução da IA

Atores: Administrador

O administrador poderá redefinir a data e hora que a IA será ativada e realizará os matches. Para ser possível a alteração, a IA não poderá estar com o status: “Em execução”, ou “Finalizado”.

Entradas:

- Nova data e hora da execução.

- Identificação do administrador responsável.
- Status atual da IA.
- Timestamp da solicitação.

Processamento desejado:

1. O Administrador, autenticado no painel, visualiza o agendamento atual [RFS14].
2. O sistema (backend) verifica o status do agendamento da IA (ex: 'Pendente', 'Em Execução', 'Finalizado').
3. Se o status for "Em Execução" ou "Finalizado", a opção de modificar é desabilitada ou retorna erro.
4. Se o status for "Pendente", o Administrador pode inserir a "Nova Data e Hora da Execução".
5. O sistema valida o formato da nova data/hora e se ela está no futuro.
6. O sistema atualiza o registro do agendamento no banco de dados com a nova data/hora.
7. O sistema registra a modificação no log de ações críticas [RFS13] (incluindo o Admin responsável e a alteração realizada).

Saídas:

- Mensagem de confirmação da alteração (ex: "Agendamento da IA atualizado para [nova data/hora].").
- Registro do agendamento atualizado no banco de dados.
- Mensagem de erro (caso a data seja inválida ou o status não seja "Pendente").

Requisitos Técnicos:

- API segura (endpoint PUT/PATCH) para atualização do agendamento (com autenticação JWT).
- Banco de dados com controle de status do agendamento da IA ('Pendente', 'Em Execução', 'Finalizado').
- Integração com [RFS13] (Registro de logs em tabela de auditoria).
- Painel administrativo com controle de estados (para validar o status antes de permitir a ação).

Critérios de Aceitação:

- A alteração somente será permitida se o status atual do agendamento da IA for “Pendente”.
- O sistema deve validar que a nova data e hora estão no futuro antes de confirmar a atualização.
- A modificação deve ser obrigatoriamente registrada no log de ações críticas [RFS13].

Dependências:

- [RFS14] Definir Data de Execução do Match Automático (IA) (Cria o agendamento que será modificado).

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RFS16] Informar término da execução

Atores: Sistema

O sistema deverá definir informar o fim da execução dos matches e enviar um relatório para o email do administrador contendo as seguintes informações:

- Tempo de duração do match;
- Quantidade de usuários com match;
- Quantidade de usuários sem match;
- Quantidade de usuários veteranos;
- Quantidade de usuários calouros;

Entradas:

- **NA**

Processamento:

- Calcular tempo total de execução.
- Obter estatísticas da base de dados.
- Gerar relatório em formato PDF ou HTML
- Enviar email automaticamente para o administrador com o relatório anexo.
- Atualizar status da IA para “Finalizado”.

Saídas:

- Email enviado ao administrador com o relatório.
- Relatório arquivado no sistema.
- Status da IA atualizado para “Finalizado”.

Requisitos Técnicos:

- Módulo de geração de relatórios.
- Sistema de envio de emails automatizado.
- Banco de dados com métricas de execução.
- Serviço de agendamento de tarefas (cron job / worker).

Critérios de Aceitação:

- Email deve ser enviado em até 1 minuto após término da execução.
- Relatório deve conter todos os campos obrigatórios.

- Status “Finalizado” deve ser atualizado corretamente no banco.

Dependências:

- RFS07 (Match Automático).
- RFS14 (Definir data de execução da IA).

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RFS17] Contactar IA

Atores: Sistema

O sistema deverá se comunicar com a API do Chatgpt para enviar os dados e receber a resposta com os alunos associados. Para isso, será utilizado o prompt abaixo:

“Você é um especialista em analisar interesses de alunos e encontrar os pares ideais entre eles, sendo um mais experiente na universidade, e o outro, ingressante. Em anexo existem duas listas de pessoas, na lista 1 são veteranos de uma universidade, e na lista 2 são calouros. Ambos possuem uma lista de interesses e informações pessoais dos alunos. Você será responsável por criar as melhores duplas possíveis. Eles serão padrinhos (veteranos) e apadrinhados (calouros). Para isso, siga a seguinte lista de prioridades ao analisar os alunos:

1. São alunos do mesmo curso (essencial);
2. São alunos de mesmo gênero;
3. São alunos de mesma idade;
4. São alunos com os mesmos interesses;

A regra 1 é ESSENCIAL para o bom funcionamento do sistema, isso quer dizer que, se não há mais alunos do mesmo curso para unir com ele, esse aluno deve retornar sem uma dupla, mesmo que haja um aluno de outro curso também sem um parceiro.

Retorne a lista de alunos associados através de um arquivo csv, separado por vírgulas com as colunas: nome_calouro, nome_veterano.”

Entradas:

- Lista de Calouros disponíveis (contendo: Nome, Curso, Gênero, Idade/Ano Nasc., interesses).
- Lista de Veteranos disponíveis (contendo: Nome, Curso, Gênero, Idade/Ano Nasc., interesses).
- O prompt de sistema (conforme descrito acima).

Processamento desejado:

1. A rotina é disparada (conforme agendamento [RFS14]).
2. O sistema busca no banco de dados [RFS01] as listas de calouros e veteranos disponíveis.
3. O sistema formata essas listas (ex: em JSON ou texto estruturado) e as anexa ao prompt.
4. O sistema estabelece conexão com a API do ChatGPT (OpenAI).
5. O sistema envia a requisição (prompt + dados) para a API e aguarda a resposta.
6. O sistema recebe a resposta em texto plano (esperado em formato CSV).
7. O sistema faz o parse (análise) do CSV recebido para extrair os pares (nome_calouro, nome_veterano).
8. O sistema associa os nomes aos IDs dos usuários no banco de dados para criar os vínculos formais (concluindo [RFS07]).

Saídas:

- **Re**sposta da API do ChatGPT contendo o texto formatado como CSV com os pares de match.
- Vínculos de match persistidos no banco de dados (resultado final do [RFS07]).

Requisitos Técnicos:

- Integração com a API do ChatGPT (OpenAI).
- Cliente HTTP (ex: Axios, Fetch) para realizar as chamadas de API.
- Rotina de parse de CSV (para processar a resposta da IA).
- Gerenciamento seguro de chaves de API (API Key) da OpenAI.

Critérios de Aceitação:

- Todos os matches retornados pela IA devem obrigatoriamente ser de alunos do mesmo curso (conforme regra "essencial" do prompt).
- A resposta da API deve ser processada e os pares devem ser extraídos corretamente.
- A chamada de API deve ter um tempo limite (timeout, ex: 60 segundos) para evitar que o sistema fique travado.
- A resposta do chatgpt deve estar em um arquivo csv separado por vírgulas.

Dependências:

- [RFS01] Cadastrar Usuário.
- [RFS07] Match Automático.
- [RFS14] Definir Data de Execução do Match Automático.

Requisitos Não-Funcionais

[RNF01] Usabilidade

[RNF01.01] Interface Intuitiva

A interface deve ser simples, com ícones e textos claros, facilitando a navegação mesmo para usuários inexperientes.

Onboard: O sistema deve prover um *wizard* (passo a passo) de *onboarding* para novos usuários (calouros) na primeira vez que acessam, guiando-os pelas etapas essenciais e explicando o valor de preencher os interesses.

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RNF01.02] Design Responsivo

O sistema deve se adaptar automaticamente a diferentes tamanhos de tela (desktop, tablet e smartphone).

Regra (Mobile-First): O design e o desenvolvimento devem seguir a abordagem *Mobile-First*. A experiência em *viewports* de smartphone (ex: 360px de largura) é prioritária.

Critério de Aceite: Todas as funcionalidades, incluindo o chat [RF04] e o cadastro [RF01], devem ser 100% operáveis e legíveis em dispositivos móveis, sem necessidade de *zoom* horizontal.

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RNF02] Confiabilidade

[RNF02.01] Recuperação de Erros

O sistema deve ser capaz de detectar e informar erros de forma amigável ao usuário, sem perda de dados.

Mensagens Claras: Mensagens de erro de validação devem ser específicas (ex: "Este e-mail já está em uso" em vez de "Erro no formulário") e exibidas *próximas* ao campo correspondente.

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RNF02.02] Backup Automático

Os dados devem ser armazenados em backup diário para evitar perda de informações em caso de falhas.

RTO (Recovery Time Objective): Em caso de falha catastrófica, o tempo de restauração do serviço (RTO) a partir do último backup deve ser de, no máximo, **4 horas**.

Armazenamento: Os backups devem ser armazenados de forma criptografada e em local geograficamente distinto do servidor de produção (ex: S3 Glacier Deep Archive ou serviço equivalente).

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RNF03] Desempenho

[RNF03.01] Tempo de Resposta

O tempo de resposta para qualquer ação do usuário não deve ultrapassar 3 segundos em condições normais de rede.

Carregamento de Página (LCP - Largest Contentful Paint): Deve ser **< 2.5 segundos** em conexões de banda larga (desktop).

Interatividade (FID - First Input Delay): O tempo de resposta a uma interação do usuário (clique) deve ser **< 100ms**.

Resposta de API: Requisições de *backend* (ex: salvar perfil, enviar mensagem de chat) devem ter um tempo de resposta do servidor (TTFB) **< 500ms** (excluindo a latência da rede do usuário).

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RNF03.02] Suporte a Múltiplos Usuários

O sistema deve suportar, simultaneamente, pelo menos 500 usuários conectados sem degradação perceptível de desempenho.

Métrica de Concorrência: O sistema deve suportar 500 usuários *concorrentes* (realizando ações de leitura/escrita, como enviar mensagens de chat [RF04] ou atualizar perfis [RF02]) mantendo os tempos de resposta do [RNF03.01].

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RNF04] Segurança

[RNF04.01] Proteção de Dados

As senhas devem ser armazenadas de forma criptografada, e as comunicações realizadas por HTTPS.

Hashing de Senhas: Senhas NUNCA devem ser criptografadas (permitindo reversão). Elas devem ser armazenadas como **HASH** usando um algoritmo moderno, assimétrico e com *salt* (ex: **Argon2id** ou **Bcrypt**). MD5 e SHA1 são completamente inaceitáveis.

HTTPS (HSTS): O sistema deve forçar HTTPS em todas as páginas e *endpoints* de API. A política de HSTS (HTTP Strict Transport Security) deve ser implementada para evitar *downgrade attacks*.

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RNF04.02] Controle de Acesso

Cada usuário deve ter acesso apenas às informações e funcionalidades correspondentes ao seu perfil (veterano, calouro ou administrador).

RBAC (Role-Based Access Control): A implementação deve seguir um padrão RBAC.

Regra (Validação no Backend): O controle de acesso NUNCA deve ser feito apenas no *frontend* (ocultando botões). Toda requisição à API (no *backend*) deve revalidar se o usuário autenticado (ex: via token JWT) tem permissão para executar aquela ação.

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RNF05] Distribuição e Compatibilidade

[RNF05.01] Navegadores Suportados

O sistema deve ser compatível com os navegadores mais utilizados (Chrome, Firefox, Edge e Safari).

Regra (Versões): O sistema deve garantir 100% de funcionalidade nas **duas últimas versões principais** dos navegadores: Google Chrome (Desktop/Android), Mozilla Firefox, Microsoft Edge e Apple Safari (Desktop/iOS).

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RNF05.02] Integração com E-mail

O sistema deve integrar-se a serviços de e-mail (como SendGrid e Mailgun) para envio de notificações automáticas.

Serviço Transacional: O envio de e-mail (notificações [RF03.02], recuperação de senha [RF01.03]) deve ser feito através de um serviço de e-mail transacional (ex: **Amazon SES**, **SendGrid**, **Mailgun**).

Regra (Anti-Spam): O envio NUNCA deve ser feito diretamente do servidor da aplicação, pois isso é garantia de cair na caixa de spam.

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RNF06] Adequação a Padrões

[RNF06.01] Padrões de Desenvolvimento

O código deve seguir boas práticas de programação (uso de versionamento, padronização de nomenclatura e documentação interna). A linguagem padrão será o inglês.

Backend: a linguagem utilizada pelo backend será NodeJS na versão 24.11. Como framework será utilizado express para o controle de rotas.

Frontend: a linguagem utilizada no frontend será React 19.1.0.

Sobre o controle de versões, a organização das branches deverá ser feita como na imagem 1, abaixo:

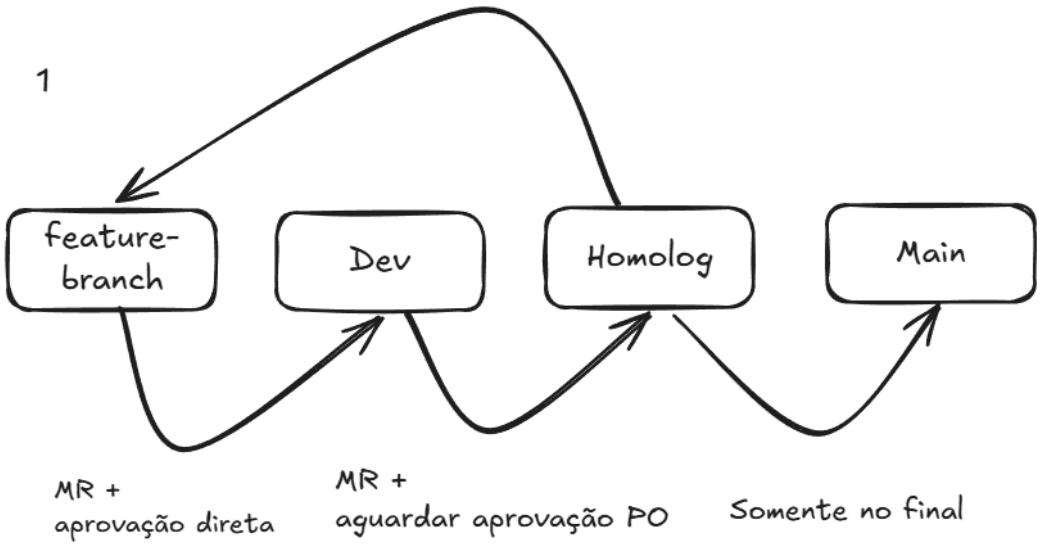

Imagem 1: organização das branches e fluxo de dados.

- A main somente será atualizada após o final do projeto (versão 1.0);
- Homolog será o ambiente onde serão realizados os testes mais completos
- Dev é o ambiente de desenvolvimento, branch onde serão analisadas as modificações para aprovar ou não, seu merge com homolog;
- feature-branch será a branch onde serão desenvolvidas as funcionalidades. São branches temporárias, que após sua integração na branch dev, serão excluídas. São provenientes sempre de homolog, onde está a versão mais estável do código durante o desenvolvimento.

Para a nomenclatura serão utilizados os seguintes padrões:

Objeto	Padrão	Exemplo
Tabelas	snake case (plural)	veteranos_table
Colunas	snake case (singular)	nome_completo
Chave Primária (PK)	snake case + id	veterano_id
Chave Estrangeira (FK)	snake case + tabela relacionada + id	veretanos_table_id

A documentação interna será feita em diversos segmentos:

Documentação do projeto: baseada em status reports, realizados semanalmente para acompanhar o projeto.

Documentação técnica: sugestões e manual através de read.me, documentação de rotas através do swagger.

Documentação de banco de dados: modelo relacional e arquivo de criação e backup das tabelas.

Prioridade:
[ ] Essencial [X] Importante [ ] Desejável

[RNF06.02] Acessibilidade

O sistema deve seguir diretrizes de acessibilidade (WCAG) para garantir que usuários com deficiências possam utilizá-lo.

Contraste: A taxa de contraste entre o texto e o fundo deve ser de no mínimo 4.5:1, conforme o padrão WCAG.

Navegação por Teclado: O sistema deve ser 100% navegável e operável utilizando apenas a tecla TAB e comandos do teclado, sem depender exclusivamente do mouse. O foco do teclado (o anel visual) deve ser sempre visível e claro.

Prioridade:
[ ] Essencial [ ] Importante [X] Desejável

[RNF07] Requisitos de Hardware e Software

[RNF07.01] Servidor

O sistema deve operar em servidor web com suporte a banco de dados relacional (MySQL ou PostgreSQL).

Banco de Dados (Motor da IA): O banco de dados relacional escolhido (MySQL ou PostgreSQL) deve ser dimensionado para suportar consultas complexas, especialmente as utilizadas pelo algoritmo de *Match* ([RF03.01]).

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

[RNF07.02] Cliente

O sistema deve funcionar em qualquer navegador sem necessidade de instalação adicional.

Zero Instalação: O sistema deve ser acessível via URL, sem a necessidade de *plugins*, *add-ons* ou qualquer *software* proprietário adicional instalado pelo usuário (além do próprio navegador).

Tecnologias Frontend: O sistema deve utilizar apenas tecnologias *open-source* padrão da web (HTML5, CSS3, JavaScript) e frameworks populares (ex: React, Vue.js ou Angular) para garantir compatibilidade e manutenibilidade.

Prioridade:
[X] Essencial [ ] Importante [ ] Desejável

Rastreabilidade

Cada requisito funcional e não funcional está identificado por um código único (RFxx ou RNFxx). Será criado um repositório no Trello onde cada card conterá o identificador com o código único, ele ligará card a funcionalidade, facilitando a rastreabilidade.

Sempre que seja necessária a introdução de alterações em relação aos requisitos descritos neste documento ou a inclusão de novos requisitos, os seguintes itens devem ser seguidos.

- Toda solicitação de mudança proveniente do Contratante deverá ser documentada por este e enviada ao gerente de projetos.
- Um documento contendo a descrição da solicitação de mudança deve ser assinado pelo gerente de projeto e cliente, formalizando assim a solicitação.
- O tempo necessário para avaliar a viabilidade técnica de uma alteração no escopo delineado nesta proposta será cobrado ao Contratante.
- A empresa fornecedora avaliará o impacto da mudança no cronograma e no custo do serviço e submeterá ao Contratante para aprovação.
- A empresa fornecedora iniciará a execução da mudança no caso de não haver impacto associado à mesma.