Demanda BitStart

IDENTIFICAÇÃO

Projeto

- BitStart - aprendendo algoritmos bit a bit

Gestor de Projeto

- Hillary Diniz Saldanha

Cliente

- Náthalee Cavalcanti / UFERSA - nathalee.almeida@ufersa.edu.br
- George Vieira/ UFERSA - george.vieira@temporarios.ufersa.edu.br

HISTÓRICO DE REGISTROS

Versão	Data	Autor	Descrição	Aprovado por
{1.0}	*06/10/2025*	*Hillary Saldanha*	Elaboração do documento	
{2.0}	*29/10/2025*	*Letícia Gonçalves*	Estabelecimento das regras de negócio	
{3.0}	*29/10/2025*	*Ana Beatriz*	Escrita dos requisitos	
{4.0}	*02/11/2025*	*Hillary Saldanha*	Escrita dos requisitos	
{5.0}	*03/11/2025*	*Rubens Alexandre*	Elaboração dos diagramas e detalhamentos de casos de uso	

SUMÁRIO

1. INTRODUÇÃO

Este documento apresenta a documentação do desenvolvimento do jogo BitStart - aprendendo algoritmos bit a bit.

2. DESCRIÇÃO DO SISTEMA

O sistema é um jogo direcionado para o auxílio do aprendizado de algoritmos e lógica de programação da disciplina de Algoritmos da Universidade Federal Rural do Semiárido (Ufersa), que após a entrevista com os clientes, Náthalee Cavalcanti (nathalee.almeida@ufersa.edu.br) e George Vieira (george.vieira@temporarios.ufersa.edu.br), professores da disciplina, onde deixa clara a necessidade de uma forma de reforço e auxílio que ajude os alunos de forma lúdica a assimilar os conteúdos vistos em sala de aula.

O jogo se passa num mundo em que a nave dos algoritmos foi “desprogramada” e o nosso protagonista Ritch (um robô) tem que ir de sala em sala consertando tudo o que está desregulado. Cada sala é temática e relacionada a um conteúdo da disciplina, pode-se comparar as salas com fases cheias de questões para serem resolvidas, e com isso, o robô irá progredindo no ajuste da nave.

Como usuários, temos os professores e os alunos. Ao se cadastrar, os professores devem adicionar suas turmas, assim, quando os alunos se cadastrarem, poderão ser vinculados à turma do professor selecionado.

3. REGRAS DE NEGÓCIO E/OU PREMISSA

Quadro 1 – **Funcionário** x **Agendamento**

Regra de Negócio	Entidades	Relação	Tipo
Um professor pode gerenciar múltiplas turmas.	Professor	Gerenciar	1:n
Uma turma é gerenciada por um único professor.	Turma		

Fonte: Autoria Própria

Quadro 2 – **Turma** x **Aluno**

Regra de Negócio	Entidades	Relação	Tipo
Uma turma pode conter múltiplos alunos.	Turma	Conter	1:n
Um aluno pode estar contido apenas em uma única turma	Aluno		

Fonte: Autoria Própria

Quadro 3 – **Aluno** x **Progresso**

Regra de Negócio	Entidades	Relação	Tipo
Um aluno possui múltiplos registros de progresso.	Aluno	Possuir	1:n
Um registro de progresso é possuído por um único aluno.	Progresso		

Fonte: Autoria Própria

Quadro 4 – **Turma** x **Relatório**

Regra de Negócio	Entidades	Relação	Tipo
Uma turma possui vários registro de relatório.	Turma	Possuir	1:n
Um registro de relatório é possuído por uma única turma.	Relatório		

Fonte: Autoria Própria

Quadro 5 – **Turma** x **Desafio**

Regra de Negócio	Entidades	Relação	Tipo
Uma turma possui vários desafios.	Turma	Possuir	1:n

Um desafio é possuído por uma única turma.	Desafio		

Fonte: Autoria Própria

3.1. Requisitos Funcionais

[RF01] — Criar e gerenciar turmas

Permitir que professores criem turmas (nome, código, semestre, disciplina, nível), editem informações, removam turmas e gerenciem matrícula (convite por código, aceitar/recusar solicitações). Deve suportar configuração de parâmetros da turma (habilitar/desabilitar ranking, visibilidade dos relatórios, idioma).

[RF02] — Cadastrar desafios e organizar por sequência lógica do conteúdo

Interface para criar/editar/excluir desafios (título, descrição, enunciado, nível de dificuldade, tags pedagógicas, código de referência, assets associados — imagens/arquivos), e definir a sequência em que os desafios aparecem nas fases/níveis da turma. Permitir agrupar desafios em módulos/sequências e rearranjar por drag-and-drop.

[RF03] — Visualizar desempenho dos alunos (nível, tentativas, tempo, acertos, erros mais comuns)

O professor pode acessar um painel analítico que mostra desempenho por aluno e por turma: nível alcançado, número de tentativas por desafio, tempo gasto por tentativa/fase, taxa de acertos/erros e ranking (se ativado). Visualizações filtráveis por período, desafio e estudante; exportáveis (ver RF04). Mostrar também estatísticas agregadas: erros mais comuns por desafio.

[RF04] — Exportar relatórios em XLSX (e opcionalmente PDF)

Permitir gerar e baixar relatórios com os dados do desempenho filtrados (por turma, aluno, intervalo). Formato principal XLSX; opção de PDF com layout de impressão. Incluir cabeçalho com metadados (turma, período, autor do relatório).

[RF05] — Ativar ou desativar score de desempenho por turma

Opção nas configurações da turma para habilitar/desabilitar cálculo e exibição de score/ranking entre alunos. Quando desativado, scores não são visíveis para alunos; logs continuam sendo gravados (configurável).

[RF06] — Registrar-se e vincular-se a uma turma

Fluxo de registro (nome, email, senha, confirmação) e opção de se vincular a uma turma via código/convite. Na vinculação, registrar papel do aluno e permitir acesso apenas ao conteúdo daquela turma.

[RF07] — Jogar as fases em ordem sequencial

Interface de jogo que apresenta fases/desafios na ordem definida pelo professor (RF02). O aluno só desbloqueia a próxima fase ao concluir critérios mínimos da atual (ex.: chegar a X% de acerto ou resolver com sucesso). Deve suportar dois modos de navegação: linear obrigatório e (opcional) desbloqueio por proficiência.

[RF08] — Ter feedback imediato em caso de erro

Enquanto joga, o sistema fornece feedback imediato e detalhado ao cometer erro: mensagem de erro, indicação de trecho incorreto (quando aplicável), sugestão didática (ex.: dica, link para tutorial) e possibilidade de tentar novamente. Feedback deve ser útil e não punitivo.

[RF09] — Consultar seu desempenho (opcional)

Área pessoal onde o aluno consulta histórico de progresso (fases concluídas, pontuação, tempo médio, atividades pendentes). Visualizações simples e explicativas, com comparativos (ex.: evolução por semana). Esse requisito pode ser marcado como opcional/parametrizável.

[RF10] — Continuar a fase do ponto salvo (salvar progresso local ou online)

Mecanismo de checkpoint para salvar progresso da fase — localmente (no navegador / arquivo) quando offline e sincronização online quando houver conexão. Permitir retomar exatamente do ponto salvo (estado do exercício, variáveis, posição). Devem existir botão “Salvar” e salvamento automático periódico.

3.1.1. Requisitos Pedagógicos

[PED01] — Cobertura de conteúdos de algoritmos

O jogo deve incluir atividades que abordem variáveis, estruturas condicionais, laços, vetores/arrays, funções/procedimentos, entrada/saída e estruturas de dados básicas. Cada conceito deve ter múltiplos níveis de complexidade.

[PED02] — Progressão alinhada à lógica de ensino

Sequência de desafios projetada para respeitar pré-requisitos (ex.: o aluno só vê laços após entender condicionais). O professor pode alterar a sequência.

[PED03] — Modo tutorial para conceitos básicos

Modo tutorial com explicações passo a passo, exemplos resolvidos, dicas interativas e exercícios guiados. Deve ser possível ativar/desativar por turma.

[PED04] — Reforço do raciocínio lógico e resolução de problemas

Desafios desenhados para incentivar decomposição de problemas, formulação de hipóteses e depuração. Inclusão de puzzles e exercícios abertos que permitam soluções alternativas.

[PED05] — Adaptação para outras disciplinas

Metadados de desafios devem conter tags de disciplina (ex.: Lógica Matemática, ED1, ED2) para facilitar reutilização e reorganização do conteúdo. Ferramenta permite importar/exportar pacotes de desafio.

3.2. Requisitos Não-Funcionais

[NFR01] — Plataformas

Aplicação compatível com Windows, Linux e Web; planejamento de compatibilidade futura para Android (progressive web app ou app nativo). O App roda em navegadores modernos em Windows/Linux; build/documentação para empacotamento Android.

[NFR02] — Modo de execução (offline/online)

Aplicação deve executar offline (recursos locais, execução de desafios) e integrar online para salvar progresso e ranking quando conectado. Teste de execução offline completo para cenários críticos; sincronização testada após reconexão.

[NFR03] — Interface (usabilidade)

Interface com opção de blocos lógicos (drag-and-drop) e modo textual simplificado (ex.: Portugol/Python). UI responsiva e acessível. Usuários conseguem resolver um desafio simples em ambos os modos em X minutos; testes de usabilidade.

[NFR04] — Performance

Rodar em máquinas de laboratório com especificações modestas (ex.: 4 GB de RAM, CPU modesta). Tempo de carga inicial aceitável (< 5s em local) e operações de UI responsivas.

[NFR05] — Segurança

Controle de acesso por login/senha; proteção básica contra XSS/CSRF; armazenamento seguro de senhas (hash+salt). Permissões: professores vs alunos.

[NFR06] — Internacionalização / Localização

Suporte a português (pt-BR) inicialmente, com possibilidade de adicionar outros idiomas.

[NFR07] — Escalabilidade do backend (opcional)

Arquitetura capaz de suportar aumento de usuários quando a função online for utilizada.

4. MODELAGEM

4.1. Diagrama de Classes

4.2. Diagrama de Caso de Uso

4.3. Diagrama Entidade-Relacionamento

4.4. Detalhamento de Caso de Uso