



UNIVERSIDADE FEDERAL DE VIÇOSA - CAMPUS
FLORESTAL

ARQUITETURA DE SOFTWARE

Product Backlog

Equipe 2º ano - Projeto Integrador

Aymê Faustino dos Santos - 4704

Florestal-MG

2024

SUMÁRIO

1. Introdução	3
2. Principais Diagramas	4
2.1. Diagrama de Casos de Uso	4
2.2. Diagrama de Classes	4
3. Documentação dos Casos de Uso	5

1. Introdução

Este documento apresenta a especificação do Product Backlog para o sistema desenvolvido pela equipe do 2º ano do Projeto Integrador 2024. O projeto consiste em um jogo educativo destinado a ensinar crianças do 2º ano do ensino fundamental sobre dispositivos e artefatos computacionais. A aplicação, gamificada, busca ajudar os Alunos a diferenciar esses dispositivos e entender em quais contextos utilizá-los.

O Product Backlog é o registro de todas as funcionalidades planejadas para o produto. Ele está sujeito a mudanças constantes e pode ser ajustado ao longo do desenvolvimento do projeto.

2. Principais Diagramas

Após o refinamento do escopo inicial do produto, foram definidos os seguintes diagramas:

2.1. Diagrama de Casos de Uso

Um modelo de caso de uso descreve como diferentes tipos de usuários interagem com o sistema, incluindo seus objetivos, as interações entre os usuários e o sistema, além do comportamento necessário para que o sistema atenda a esses objetivos. Os casos de uso no diagrama estão numerados conforme sua importância dentro do sistema. No entanto, cabe ao Líder de Projetos decidir a melhor ordem para sua execução.

Durante a redução do escopo inicial, a equipe decidiu remover os casos de uso relacionados à escola, mantendo apenas aqueles voltados para o Aluno.

As próximas seções fornecem esclarecimentos mais detalhados sobre os casos de uso presentes no diagrama. No entanto, futuramente, esses casos serão descritos com maior profundidade, a fim de auxiliar os desenvolvedores na correta implementação das funcionalidades apresentadas.

2.2. Diagrama de Classes

Um modelo de classes é fundamental para representar os objetos que compõem o sistema, mostrar os relacionamentos entre eles e definir suas responsabilidades e serviços. Já um diagrama de classes em nível de especificação oferece uma visão detalhada da estrutura do sistema, auxiliando na comunicação entre desenvolvedores, analistas e designers. Esses diagramas demonstram o funcionamento geral do sistema, mas serão posteriormente detalhados para que os desenvolvedores possam implementar corretamente as funcionalidades de cada componente do produto.

```

classDiagram
    class DAOAluno {
        <<boundary>>
        -selecionar(filtro: Object): selectQuery
        -adicionar(aluno: Aluno): boolean
        -remover(aluno: Aluno): boolean
    }
    class DAOPartida {
        -selecionar(filtro: Object): selectQuery
        -adicionar(partida: Partida): boolean
    }
    class DAO {
        -conexao: MySQL
    }
    class ControladorPontuacao {
        -btn_prosseguir: Button
        +prosseguir(): void
    }
    class ControladorPontuacaoFinal {
        -terminarjogo: Button
    }
    class ControladorLogin {
        -box_inputAluno: String
        -btn_titulo: Button
        +toTitulo(): void
    }
    class ControladorTitulo {
        -btn_jogar: Button
        +jogar(): void
    }
    class ControladorFase {
        -btn_prosseguir: Button
        +prosseguir(): void
    }
    class Partida {
        -pontuacaoTotal: int
        -faseAtual: char
        -tempoTotal: int
    }
    class Aluno {
        -nome: char
        -turma: int
    }
    class Nivel {
        <<abstract>>
        -pontuacao: int
        -tempo: int
        -concluido: boolean
        -erros: int
        -dicas: ArrayList<String>
        -erroMin: int
        -enunciado: String
    }
    class NivelCarta {
    }
    class NivelContexto {
        -imagemContexto: String
        -spices: ArrayList<String>
        -idOpcaoCorreta: int
    }
    class Carta {
        -imagem: String
        -id: int
        -idCorrespondente: int
    }
    class TelaPontuacao {
        -controladorPontuacao: ControladorPontuacao
    }
    class TelaPontuacaoFinal {
        -controladorPontuacaoFinal: ControladorPontuacaoFinal
    }
    class TelaLogin {
        -controladorLogin: ControladorLogin
    }
    class TelaTitulo {
        -controladorTitulo: ControladorTitulo
    }
    class TelaFase {
        -controladorFase: ControladorFase
    }
    class TelaFaseCartas {
    }
    class TelaFaseContexto {
    }
    class TelaIntroducaoFase {
    }

    DAOAluno <|-- DAO
    DAOAluno <|-- DAOPartida
    DAOAluno <|-- DAO
    ControladorPontuacao <|-- ControladorPontuacaoFinal
    ControladorPontuacao <|-- ControladorLogin
    ControladorPontuacao <|-- ControladorTitulo
    ControladorPontuacao <|-- ControladorFase
    Partida <|-- Aluno
    Nivel <|-- NivelCarta
    Nivel <|-- NivelContexto
    Nivel <|-- Carta
    TelaPontuacao <|-- TelaPontuacaoFinal
    TelaPontuacao <|-- TelaLogin
    TelaPontuacao <|-- TelaTitulo
    TelaPontuacao <|-- TelaFase
    TelaFaseCartas <|-- TelaFaseContexto
    TelaIntroducaoFase <|-- TelaFase

    DAOAluno "1" -- "1" ControladorPontuacao
    DAOAluno "1" -- "1" ControladorPontuacaoFinal
    DAOAluno "1" -- "1" ControladorLogin
    DAOAluno "1" -- "1" ControladorTitulo
    DAOAluno "1" -- "1" ControladorFase
    DAOPartida "1" -- "1" Partida
    DAO "1" -- "1" Partida
    ControladorPontuacao "1" -- "1" Partida
    ControladorPontuacaoFinal "1" -- "1" Partida
    ControladorLogin "1" -- "1" Aluno
    ControladorTitulo "1" -- "1" Aluno
    ControladorFase "1" -- "1" Partida
    ControladorFase "1" -- "1" NivelCarta
    ControladorFase "1" -- "1" NivelContexto
    ControladorFase "1" -- "1" Carta
    Partida "1" -- "1" Aluno
    Aluno "1" -- "1" Partida
    NivelCarta "1" -- "1" NivelContexto
    NivelCarta "1" -- "1" Carta
    Carta "1" -- "1" Carta
    TelaPontuacao "1" -- "1" ControladorPontuacao
    TelaPontuacaoFinal "1" -- "1" ControladorPontuacaoFinal
    TelaLogin "1" -- "1" ControladorLogin
    TelaTitulo "1" -- "1" ControladorTitulo
    TelaFase "1" -- "1" ControladorFase
    TelaFaseCartas "1" -- "1" TelaFaseContexto
    TelaIntroducaoFase "1" -- "1" TelaFase
  
```

The diagram illustrates the architecture of a game system, organized into several layers and components:

- DAO Layer (Data Access Objects):**
 - DAOAluno** (Boundary): Manages student data with methods `selecionar(filtro: Object)`, `adicionar(aluno: Aluno)`, and `remover(aluno: Aluno)`.
 - DAOPartida** (Boundary): Manages game data with methods `selecionar(filtro: Object)` and `adicionar(partida: Partida)`.
 - DAO**: A base DAO class with a `conexao: MySQL` attribute.
- Controlador Layer (Controllers):**
 - ControladorPontuacao** (Boundary): Contains `btn_prosseguir: Button` and a `prosseguir(): void` method.
 - ControladorPontuacaoFinal** (Boundary): Contains a `terminarjogo: Button`.
 - ControladorLogin** (Boundary): Contains `box_inputAluno: String`, `btn_titulo: Button`, and a `toTitulo(): void` method.
 - ControladorTitulo** (Boundary): Contains `btn_jogar: Button` and a `jogar(): void` method.
 - ControladorFase** (Boundary): Contains `btn_prosseguir: Button` and a `prosseguir(): void` method.
- Entity Layer (Entities):**
 - Partida** (Boundary): Attributes include `pontuacaoTotal: int`, `faseAtual: char`, and `tempoTotal: int`.
 - Aluno** (Boundary): Attributes include `nome: char` and `turma: int`.
 - Nivel** (Abstract Boundary): Attributes include `pontuacao: int`, `tempo: int`, `concluido: boolean`, `erros: int`, `dicas: ArrayList<String>`, `erroMin: int`, and `enunciado: String`.
 - NivelCarta** (Boundary): Inherits from **Nivel**.
 - NivelContexto** (Boundary): Inherits from **Nivel**. Attributes include `imagemContexto: String`, `spices: ArrayList<String>`, and `idOpcaoCorreta: int`.
 - Carta** (Boundary): Attributes include `imagem: String`, `id: int`, and `idCorrespondente: int`.
- Tela Layer (Views/Interfaces):**
 - TelaPontuacao** (Boundary): Contains a `controladorPontuacao: ControladorPontuacao`.
 - TelaPontuacaoFinal** (Boundary): Contains a `controladorPontuacaoFinal: ControladorPontuacaoFinal`.
 - TelaLogin** (Boundary): Contains a `controladorLogin: ControladorLogin`.
 - TelaTitulo** (Boundary): Contains a `controladorTitulo: ControladorTitulo`.
 - TelaFase** (Boundary): Contains a `controladorFase: ControladorFase`.
 - TelaFaseCartas** (Boundary): Inherits from **TelaFase**.
 - TelaFaseContexto** (Boundary): Inherits from **TelaFase**.
 - TelaIntroducaoFase** (Boundary): Inherits from **TelaFase**.

Relationships:

- DAOAluno** is associated with **ControladorPontuacao**, **ControladorPontuacaoFinal**, **ControladorLogin**, **ControladorTitulo**, and **ControladorFase**.
- DAOPartida** is associated with **Partida**.
- DAO** is associated with **Partida**.
- ControladorPontuacao** is associated with **Partida**.
- ControladorPontuacaoFinal** is associated with **Partida**.
- ControladorLogin** is associated with **Aluno**.
- ControladorTitulo** is associated with **Aluno**.
- ControladorFase** is associated with **Partida**, **NivelCarta**, **NivelContexto**, and **Carta**.
- Partida** is associated with **Aluno**.
- Aluno** is associated with **Partida**.
- NivelCarta** is associated with **NivelContexto** and **Carta**.
- Carta** is associated with **Carta** (self-association).
- TelaPontuacao** is associated with **ControladorPontuacao**.
- TelaPontuacaoFinal** is associated with **ControladorPontuacaoFinal**.
- TelaLogin** is associated with **ControladorLogin**.
- TelaTitulo** is associated with **ControladorTitulo**.
- TelaFase** is associated with **ControladorFase**.
- TelaFaseCartas** and **TelaFaseContexto** inherit from **TelaFase**.
- TelaIntroducaoFase** inherits from **TelaFase**.

3. Documentação dos Casos de Uso

Os Diagramas de Caso de Uso facilitam o entendimento de como o sistema funciona, contudo nem sempre são o suficiente para apresentar detalhes do funcionamento. Por esse motivo, a seguir estão apresentadas as documentações dos casos de uso presentes no diagrama da Seção 2.2.

A documentação de um Caso de Uso descreve por meio de uma linguagem simples, sua função em linhas gerais e define o passo a passo que o usuário realizará na funcionalidade que está sendo descrita.

Iniciar jogo (CSU01)

Sumário: O Aluno utiliza o sistema para se cadastrar e iniciar o jogo.

Ator primário: Aluno

Pré-Condições: O Aluno deve estar logado no sistema.

Fluxo Principal:

- 1) O sistema exibe a tela inicial com a opção de “Jogar”.
- 2) O Aluno seleciona a opção “Jogar”.
- 3) O sistema exibe a tela de identificação com o campo “nome” e “turma”.
- 4) O Aluno preenche o campo “nome”.
- 5) O Aluno preenche o campo “turma”
- 6) O sistema valida o nome preenchido.
- 7) O sistema valida a turma preenchida.

Fluxo Alternativo (5): Nome inválido

- a. O sistema exibe uma mensagem e solicita um novo preenchimento.
- b. O Sistema valida e retorna ao passo 6 do fluxo principal.

Regras de Negócio: RN1

Requisitos: RF1

Jogar fase 1 (CSU02)

Sumário: O Aluno utiliza o sistema para jogar a primeira fase do jogo.

Ator primário: Aluno

Pré condições: O Aluno precisa estar logado no sistema.

Fluxo Principal:

- 1) O Aluno inicia a fase 1.
- 2) O sistema mostra na tela as cartas com as imagens dos dispositivos e começa a cronometrar o tempo que o Aluno irá demorar até concluir o nível.
- 3) O Aluno tenta relacionar os dispositivos idênticos corretamente.
- 4) O sistema mostra o nome do dispositivo, utilidades e a explicação sobre os dispositivos relacionados corretamente.
- 5) O Aluno relaciona todas as cartas do nível atual corretamente.
- 6) O estudante inicia o próximo nível.
- 7) O fluxo segue do mesmo modo que entre os passos 2 a 6, até o último nível.
- 8) O sistema se encerra e registra as informações até o último nível completado no banco de dados.

Fluxo Alternativo (3): Relacionando as cartas de forma errada

Ao relacionar erroneamente, o sistema mostra que a relação está errada e o sistema volta para o passo 3.

Ao errar três ou mais vezes seguidas, o sistema mostra uma dica e volta para o passo 3.

Fluxo de Exceção (3): Violação da RNx

Se o Aluno encerrar o jogo antes de terminar o nível atual, o sistema se encerra.

Pós-condições: O Aluno terminou a fase e as informações foram salvas no banco de dados.

Regras de Negócio: RN1, RN2, RN3, RN6, RN8, RN9, RN11.

Requisitos: RF4, RF9, RF12.

Jogar Fase 2 (CSU03):

Sumário: Aluno usa o sistema para jogar os níveis da fase 2.

Ator Primário: Aluno.

Atores Secundários: Gerenciador de Banco de Dados.

Precondições: O Aluno deve ter concluído a fase 1 com sucesso.

Fluxo Principal:

1. O Aluno inicia a fase 2.
2. O sistema mostra na tela as cartas com as imagens dos dispositivos e começa a cronometrar o tempo que o Aluno demorará até concluir o nível.
3. O Aluno tenta relacionar os dispositivos semelhantes corretamente.
4. O sistema mostra os nomes dos dispositivos, semelhanças, utilidades e a explicação sobre os dispositivos relacionados corretamente.
5. O Aluno relaciona todas as cartas do nível atual corretamente.
6. O estudante inicia o próximo nível.
7. O fluxo segue do mesmo modo que entre os passos 2 a 6, até o último nível.
8. O sistema se encerra e registra as informações até o último nível completado no banco de dados.

Fluxo Alternativo (3): Relacionando as cartas de forma errada

- a. Ao relacionar erroneamente, o sistema mostra que a relação está errada e o sistema volta para o passo 3.
 - i. Ao errar três ou mais vezes seguidas, o sistema mostra uma dica e volta para o passo 3.

Fluxo de Exceção (3): Violação da RNx

- a. Se o Aluno encerrar o jogo antes de terminar o nível atual, o sistema se encerra.

Pós-condições: O Aluno terminou a fase e as informações foram salvas no banco de dados.

Regras de Negócio: RN1, RN2, RN4, RN6, RN8, RN9, RN11.

Requisitos: RF5, RF9, RF12.

Jogar Fase 3 (CSU04):

Sumário: O Aluno usa o sistema para jogar os níveis da fase 3.

Ator Primário: Aluno.

Atores Secundários: Gerenciador de Banco de Dados.

Precondições: O Aluno deve ter concluído a fase 2 com sucesso.

Fluxo Principal:

1. O Aluno inicia a fase 3.
2. O sistema mostra na tela uma situação, uma pergunta e opções de dispositivos.
3. O Aluno tenta selecionar o dispositivo que responde a pergunta corretamente.
4. O sistema mostra porque esse dispositivo é o correto.
5. O Aluno termina a fase e o caso de uso se encerra
6. O estudante inicia o próximo nível.
7. O fluxo segue do mesmo modo que entre os passos 2 a 6, até o último nível.
8. O sistema se encerra e registra as informações até o último nível completado no banco de dados.

Fluxo Alternativo (3): Responde a pergunta de forma errada

- a. Ao responder erroneamente, o sistema mostra por que a alternativa está errada.
- b. O sistema mostra uma dica e volta para o passo 3.

Fluxo de Exceção (3): Violação da RNx

- a. Se o Aluno encerrar o jogo antes de terminar o nível atual, o sistema se encerra.

Pós-condições: O Aluno terminou a fase e as informações foram salvas no banco de dados.

Regras de Negócio: RN1, RN2, RN5, RN7, RN8, RN10, RN11.

Requisitos: RF6, RF9, RF12.