

Feijão Mágico

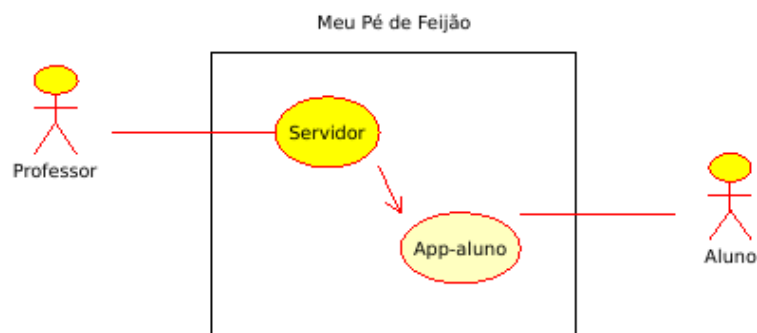
OBJETIVO DO PROJETO: desenvolver uma *game* educativo para alunos na faixa de idade entre 9 e 10 anos.

Objetivo do game: obter a maior pontuação ao final do ciclo de vida do pé de Feijão. A pontuação será definida por:

- Produção final do pé de feijão [indicada pela **força**];
- **Estrelinhas** não utilizadas ou acumuladas (explicações abaixo).

Atores envolvidos:

- Professor: responsável por criar um jogo (desafio)
- Alunos: turma de aluno(s) que acessa o jogo definido pelo professor



O módulo **servidor** é um site onde o professor realiza as seguintes ações:

- Registro de dados: cadastro dos dados do professor (criando um usuário);
- Criar um jogo: o professor seleciona uma disciplina e o conteúdo (da disciplina) que deseja abordar no jogo, o servidor gera um código para o jogo (chave de acesso);
- Apresentar estatísticas sobre o jogo: ao final do período do jogo, são fornecidas estatísticas sobre a turma e sobre cada aluno (questões respondidas certas e erradas)
- apresentar a pontuação de cada participante no jogo - e o(s) vencedor(es)

O módulo **App-aluno** é o aplicativo instalado no celular do aluno, responsável pelas ações:

- Acessar o jogo definido pelo professor: responder as questões e ao final do período informar sua pontuação ao servidor.

Regras do jogo

Definições:

- Tempo de crescimento da planta: 6 dias (o resultado é apresentado após o 6º dia)
- A planta apresenta um indicador denominado **força**: percentual que indica o nível de saúde da planta:
 - Valor inicial de **força**: 100%
 - O valor de **força** será mantido se aluno executar as tarefas diárias (regar a planta).

- O valor de **força** será reduzida se necessidades diárias da planta não são atendidas (apresentadas abaixo).

*** Cada rega poderá ser feita uma única vez durante o período de 24h, e será habilitada somente após o aluno responder a uma questão. Se a resposta estiver (Penalidade = 18):

- Correta na primeira tentativa: a planta permanece com a mesma força;
- correta na segunda tentativa: a planta reduz sua força em Penalidade/2;
- Incorreta na segunda tentativa: a planta reduz sua forma em Penalidade (penalidade menor se houver tentativa por parte do aluno ?????);

Se aluno não realizar rega da planta, será penalizado em 18.

** Um recurso especial: ESTRELINHAS (ou ADUBO)

Cada aluno recebe 0 (zero) unidades de estrelinhas no início do game.

O aluno pode aumentar a quantidade de estrelinhas ao responder corretamente uma pergunta.

Serão permitidas até 9 questões por dia. Pontuação por questão:

- Acerto na primeira tentativa: 2 estrelinhas;
- Acerto na segunda tentativa: 1 estrelinha.
- Erro na segunda tentativa: 0 estrela na questão (e a questão é encerrada).

As estrelinhas podem ser utilizadas para aumentar a força do pé de feijão, cada estrelinha (unidade) permite a recuperação de 1 ponto na força do pé de feijão. A força do pé de feijão não ultrapassa 100.

Após o período do jogo será computada a pontuação do aluno, por exemplo:

- Força do pé de feijão: 96 (F)
- Estrelinhas restante: 20 unidades (E)

Resultado final (pontuação final): (F + E)

Exemplo dos cuidados diários com o pé de feijão:

- Atualização da **força** do pé de feijão.
- Valor inicial de **força**: 100
- Penalidade: valor máximo de desconto na **força** atual (as taxas de penalidades devem ser distribuídas conforme tabela abaixo por dia)

Algoritmo para atualização da força (:

```
tentativa = 0
while tentativa < 2
    leia resposta
    if resposta correta
        break
    tentativa++
fimWhile
```

AtualizarForcaFeijao(tentativa) //chamada a função AtualizarForcaFeijao

=====

```
AtualizarForcaFeijao(int tentativa)
{
    Penalidade = 18
```

```

if tentativa = 1          // acerto na 2ª tentativa – reduz metade de penalidade
    Penalidade = Penalidade/2
else
    if tentativa = 0      //acerto na 1ª tentativa não ocorre penalidade e a força permanece a mesma
        Penalidade =0

}
=====

```

Relação entre imagem e a força do feijão:

- **Penalidade por não regar => redução de 18 na força**
- Força inicial: 100

Dia	1	2	3	4	5	6
Regar	--	OK	1/2	OK	OK	--
Penalidade	18	0	9	0	0	7
ESTRELINHA	10	10	5	7**	7	0
Força (final)	82	82	78	83	83	72
imagem	Anormal 1	anormal1	anormal 1	Anormal 1	Anormal 1	Anormal 1

** usou 5 estrelinhas e ganhou outras 7 respondendo perguntas

As cores na tabela representa as 5 fases do ciclo de vida do feijão.

Cada fase apresenta 5 **imagens**:(utilizadas pelo visualizador):

1. **Crescimento normal:** força => 90%
2. **Crescimento anormal 1:** 70%<=força<90%
3. **Crescimento anormal 2:** 50%<=força<70%
4. **Crescimento anormal 3:** 30%<=força<50%
5. **Muito fraca:** 1%<=força<30%
6. **Morte da planta:** força=0%

A imagem será selecionada de acordo com o percentual de crescimento da planta.

Casos de Uso

1. Configuração Inicial

1.1. Objetivo

Informar dados do usuário (aluno)

1.2. Ator

Aluno.

1.3. Fluxo Principal

P1. Exibir tela inicial do aplicativo (protótipo de tela)

P2. Leitura do nome do aluno

P3. Gravar nome do aluno (no dispositivo local) – use um arquivo de configuração **config.json**

formato do arquivo:

```
{  
  "nome": "fulano de tal"  
}
```

1.4. Fluxos Alternativos

A1. Usuário selecionar botão INFO: abrir tela com texto explicativo sobre o game.

1.5. Fluxos de Exceção

E1. Não aceitar nome vazio

1.6. Pré-condições

Aplicativo instalado no dispositivo móvel do aluno.

Executado uma única vez.

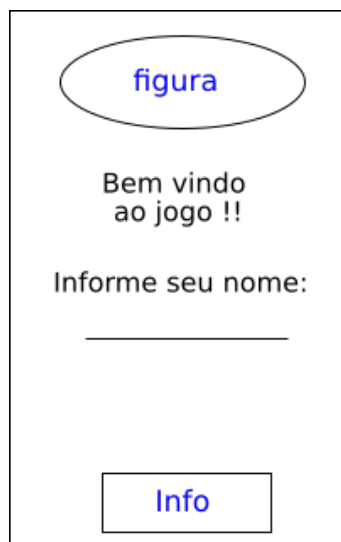
1.7. Pós-condições

Habilitar o início de um game.

1.8. Casos de Teste

Digitar o nome do aluno e verificar se foi inserido no arquivo de configuração local o nome do aluno.

1.9. Protótipo de Tela



A wireframe of a game screen prototype. It features a rectangular frame containing the following elements from top to bottom: an oval button labeled "figura" in blue text; a welcome message "Bem vindo ao jogo !!"; a prompt "Informe seu nome:" followed by a horizontal input line; and a rectangular button labeled "Info" in blue text.

figura

Bem vindo
ao jogo !!

Informe seu nome:

Info

2. Selecionar o Game

2.1. Objetivo

Selecionar um jogo para executar as atividades.

2.2. Ator

Aluno

2.3. Fluxo Principal

- P1. Abrir arquivo `gamesdata.json`
- P2. Carregar variável com `Nome_Fantasia` de todos os games
- P3. Exibir tela no aplicativo (protótipo de tela) a lista de jogos cadastrados. Os jogos encerrados (que excedem o período de validade do game) devem aparecer em cor diferenciada.
- P4. Leitura do game selecionado (por click na opção desejada).
 - P4.1 Carregar dados do game selecionado do arquivo `gamesdata.json`:
 - P4.2 Executar Caso de USO `Atualizar Força da Planta`
- P5. Abrir caso de uso INICIAR ATIVIDADE DO GAME.

2.4. Fluxos Alternativos

- A1. Aluno seleciona NOVO: ativar caso de uso NOVO GAME
- A2. Aluno seleciona SAIR: fechar aplicativo do aluno.

2.5. Fluxos de Exceção

E1. Os jogos já concluídos (prazo encerrado) devem aparecer em cor opaca.

E2. Se o aluno selecionar um jogo concluído: será apresentada a pontuação obtida no jogo.

2.6. Pré-condições

2.7. Pós-condições

Abrir caso de uso INICIAR ATIVIDADE DO GAME.

2.8. Casos de Teste

2.9. Protótipo de Tela

figura

Olá [nome]

Selecione o Game:

Novo Sair

Exemplo de arquivo **games.json** (use o arquivo que está no github).

```
{
  "jogos": [
    {
      "codigo": "As12qw",
      "nome_fantasia": "Equações",
      "disciplina": "Matemática",
      "professor": "Nome Prof",
      "datainicio": "2022-02-24",
      "datafim": "2022-03-01",
      "força": "90",
      "data_atualização_força": "2022-03-26",
      "qtd_estrelinhas": "12"
    },
    {
      "codigo": "4rASx",
      "nome_fantasia": "Bacias Hidrográficas",
      "disciplina": "Geografia",
      "professor": "Nome Prof",
      "datainicio": "2022-02-24",
      "datafim": "2022-03-01",
      "força": "90",
      "data_atualização_força": "2022-03-26",
      "qtd_estrelinhas": "12"
    },
    {
      "codigo": "A12B3",
      "nome_fantasia": "História do Brasil",
      "disciplina": "História",
      "professor": "Nome Prof",
      "datainicio": "2022-02-24",
      "datafim": "2022-03-01",
      "força": "90",
      "data_atualização_força": "2022-03-26",
      "qtd_estrelinhas": "12"
    }
  ]
}
```

}¹

3. Atualizar Força da Planta

3.1. Objetivo

Executar tarefa de atualizar força da planta.

3.2. Ator

Sistema.

3.3. Fluxo Principal

Valor para tentativa:

- 00: Questão iniciada mas não respondida

P1. Abrir arquivo: **questoes_respondidas_[codigo do game].json** - carregar vetor com dados das questões respondidas: [id_questao] com tipo == "P" e tentativa == "00"

Enquanto Existe questão com tentativa == 00 faça

gravar no arquivo **questoes_respondidas_[codigo do game].json** no registro
id_questao == id_questão e tipo = "P"

tentativa = 21

atualizar (o arquivo gamesdata.json) em regitro "codigo" == codigo do game

força = força – 18

fimWhile

3.4. Fluxos Alternativos

3.5. Fluxos de Exceção

3.6. Pré-condições

3.7. Pós-condições

3.8. Casos de Teste

3.9. Protótipo de Tela

Não existe tela.

4. Novo Game

4.1. Objetivo

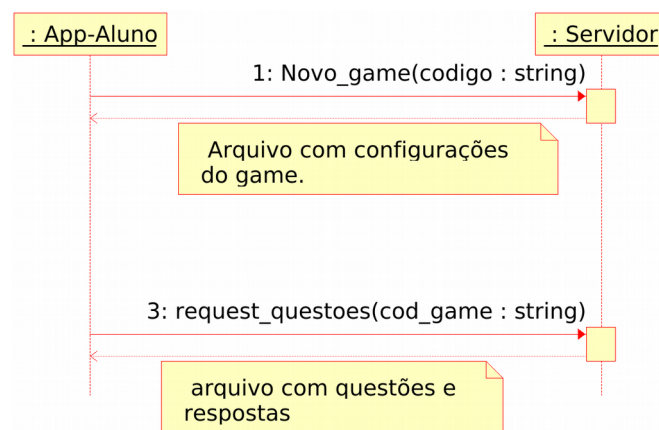
Configurar uma atividade (iniciar um game).

4.2. Ator

Aluno

4.3. Fluxo Principal

- P1. Exibir tela no aplicativo (protótipo de tela), para leitura do código informado pela professora.
- P2. Leitura do código da atividade (game)
- P3. Enviar código da atividade para servidor remoto. O servidor deve enviar arquivo json com configurações do game.
- P4. App requisita arquivo com questões.



Formato do arquivo de configuração (com valores de exemplo):

```
{
  "codigo": "Aw3erf",
  "nome_fantasia": "Frações",
  "disciplina": "Matemática",
  "professor": "Nome Prof",
  "datainicio": "2022-02-24",
  "datafim": "2022-03-01",
  "força": "100",
  "data_atualização_força": "2022-03-24",
  "qtd_estrelinhas": "0"
}
```

**** Inserir os dados acima (recebidos) no arquivo `games.json` no dispositivo móvel. [veja exemplo**

```

abaixo]
{
  "jogos": [
    {
      "codigo": "As12qw",
      "nome_fantasia": "Equações",
      "disciplina": "Matemática",
      "professor": "Nome Prof",
      "datainicio": "2022-02-24",
      "datafim": "2022-03-01",
      "força": "90",
      "data_atualização_força": "2022-03-26",
      "qtd_estrelinhas": "12"
    },
    {
      "codigo": "4rASx",
      "nome_fantasia": "Bacias Hidrográficas",
      "disciplina": "Geografia",
      "professor": "Nome Prof",
      "datainicio": "2022-02-24",
      "datafim": "2022-03-01",
      "força": "90",
      "data_atualização_força": "2022-03-26",
      "qtd_estrelinhas": "12"
    },
    {
      "codigo": "A12B3",
      "nome_fantasia": "História do Brasil",
      "disciplina": "História",
      "professor": "Nome Prof",
      "datainicio": "2022-02-24",
      "datafim": "2022-03-01",
      "força": "90",
      "data_atualização_força": "2022-03-26",
      "qtd_estrelinhas": "12"
    }
  ]
},
{
  "codigo": "Aw3erf",
  "nome_fantasia": "Frações",
  "disciplina": "Matemática",
  "professor": "Nome Prof",
  "datainicio": "2022-02-24",
  "datafim": "2022-03-01",
  "força": "100",
  "data_atualização_força": "2022-03-24",
  "qtd_estrelinhas": "0"
}

```

]

P4. O servidor envia arquivo com as perguntas relativas ao conteúdo selecionado pelo professor. Gravar arquivo com nome `questoes_[codigo do game].json` (exemplo: `questoes_AA2209.json`)

P4.1 Ativar Caso de Uso: **Preparar Questões**

P5. Habilitar o início do game

4.4. Fluxos Alternativos

Não se aplica.

4.5. Fluxos de Exceção

E1. Servidor sem conexão

Exibir mensagem de erro de conexão

E2 Perda de parte dos dados enviados pelo servidor

Exibir mensagem de erro e descartar dados recebidos. Retornar para P3.

4.6. Pré-condições

Dados do usuário informados.

4.7. Pós-condições

Conjunto de atividades habilitadas ao aluno.

4.8. Casos de Teste

Enviar código ao servidor e verificar a resposta recebida.

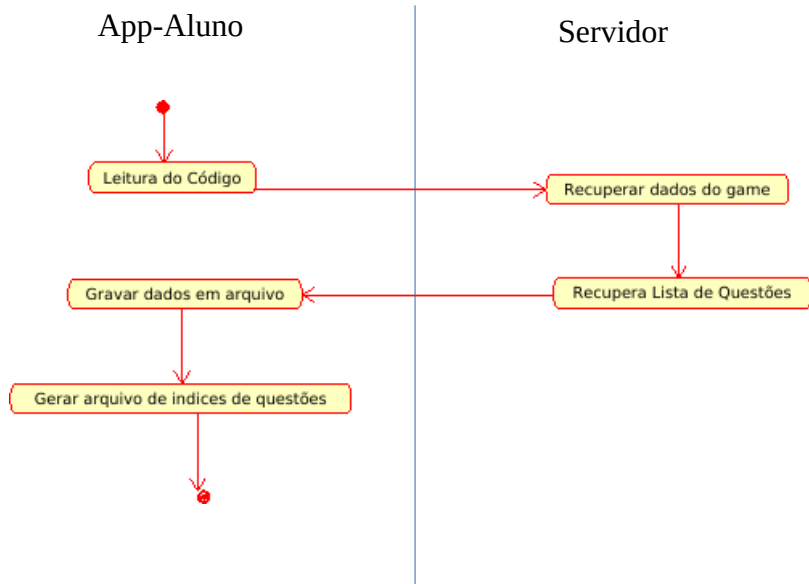
4.9. Protótipo de Tela

figura

Informe o código do novo game:

OK

Retornar



5. Caso de Uso: Preparar Questões

5.1. Objetivo

Preparar questões para uso.

5.2. Ator

5.3. Fluxo Principal

P1. Abrir arquivo de questões: questoes_[codigo do game].json ([questoes_AA2209.json](#) disponível no github).

Exemplo de formato:

```
{
  "codigo": "AA2209",
  "questao": [
    {
      "id_questao": "90",
      "resp_correta": "02",
      "comentario": "texto de ajuda",
      "respostas": [
        {
          "id_resp": "01",
          "texto": "resposta 01"
        },
        {
          "id_resp": "02",
          "texto": "resposta 02"
        },
        {
          "id_resp": "03",
          "texto": "resposta 03"
        },
        {
          "id_resp": "04",
          "texto": "resposta 04"
        }
      ],
    },
  ],
}

{
  "id_questao": "10",
  "resp_correta": "01",
  "comentario": "texto de ajuda",
  "respostas": [
    {
```

```

        "id_resp": "01",
        "texto": "resposta 10-01"
    },
    {
        "id_resp": "10-02",
        "texto": "resposta 02"
    },
    {
        "id_resp": "10-03",
        "texto": "resposta 03"
    },
    {
        "id_resp": "10-04",
        "texto": "resposta 04"
    }
],
}
]
}

```

P2. Gerar arquivo `questoes_disponiveis_[codigo game].json` (`questoes_disponiveis_AA2209.json`) com o seguinte formato:

```

questoes_disponiveis_AA2209.json
{
  "codigo": "AA2209",
  "questao": [
    {
      "id_questao": "92"
    },
    {
      "id_questao": "80"
    },
    { "id_questao": "90"
    },
    { "id_questao": "10"
    }
  ]
}

```

O arquivo apresenta o código das questões do game disponíveis para uso durante as atividades de **rega** e obter **estrelinhas**. A ordem das questões (valor de "id_questão") deve ser aleatória.

5.4. Fluxos Alternativos

5.5. Fluxos de Exceção

5.6. Pré-condições

5.7. Pós-condições

Verificar se arquivo `questoes_disponiveis_[codigo game].JSON` existe no dispositivo móvel e se as questões estão em ordem aleatória.

Seção Iniciar Atividade do Game

6. Iniciar Atividade do Game

6.1. Objetivo

Exibir imagem da condição atual do pé de feijão e conjunto de opções para o game.

6.2. Ator

Aluno

6.3. Fluxo Principal

P1. Obter a força da planta (dos dados do game)

P2. Exibir tela com imagem da planta (equivalente a sua força)

A imagem depende da **força** momentânea do pé de feijão:

1. **Crescimento normal:** $\text{força} \Rightarrow 90\%$
2. **Crescimento anormal 1:** $70\% \leq \text{força} < 90\%$
3. **Crescimento anormal 2:** $50\% \leq \text{força} < 70\%$
4. **Crescimento anormal 3:** $30\% \leq \text{força} < 50\%$
5. **Muito fraca:** $1\% \leq \text{força} < 30\%$
6. **Morte da planta:** $\text{força} = 0\%$

P3. Exibir dados da planta: força e quantidade de estrelinhas

P4. Ativar caso de uso de acordo com seleção do usuário (regar, obter estrelas, usar estrelas, sair)

6.4. Fluxos Alternativos

6.5. Fluxos de Exceção

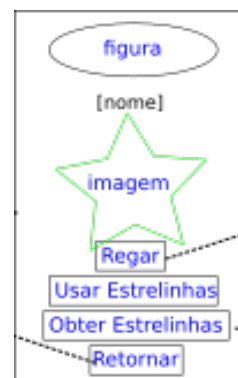
6.6. Pré-condições

Game ainda não expirou o período.

6.7. Pós-condições

6.8. Casos de Teste

6.9. Protótipo de Tela



7. Regar Planta

7.1. Objetivo

Executar tarefa de regar a planta.

7.2. Ator

Aluno

7.3. Fluxo Principal

Valores para tentativa {primeiro dígito: tentativa – segundo dígito: resultado}:

- 10: primeira tentativa resposta errada
- 11: primeira tentativa resposta correta
- 20: segunda tentativa resposta errada
- 21: segunda tentativa resposta correta

P1. No arquivo: **questoes_respondidas_[codigo do game].json** - carregar vetor com dados das questões respondidas [data, id_questao, tentativa] com tipo == "P" {"P" para plantas}

Verificar se existem questões com data == data de hoje (data do sistema) e (tentativa == 10 ou tentativa == 00)

Continuar = verdadeiro

Enquanto continuar faça

Se não existe então // não existe questao com data de hoje e tentativa == "00" ou tentativa == "10"

buscar questão no arquivo de questões: **questoes_disponiveis_[codigo do game].json**

selecione o primeiro registro (armazenar valor de *id_questão*) – remover o valor do arquivo

questoes_disponiveis_[codigo do game].json

criar registro no arquivo **questoes_respondidas_[codigo do game].json** com:

data = data de hoje

id_questao = *id_questão*

tentativa = 00 // estado inicial

tipo = "P"

Ativar Caso de USO **Exibir Questão** (id_questao)

Se resposta correta então

gravar no arquivo **questoes_respondidas_[codigo do game].json** no registro **id_questao == *id_questão*** e data = data de hoje e tipo = "P"

tentativa = 11

// força se mantem

continuar = falso

senão // primeira resposta incorreta

gravar no arquivo **questoes_respondidas_[codigo do game].json** no registro **id_questao == *id_questão*** e data = data de hoje e tipo = "P"

tentativa = 10

força = força – 9 // gravar no arquivo json

Exibir pop up Perguntando se deseja continuar novamente

fimSE

senão // existe

```

continuar = falso
Ativar Caso de USO Exibir Questão (id_questao)
Se resposta correta então
    Se (tentativa == 00)
        gravar no arquivo questoes_respondidas_[codigo do game].json no
        registro id_questao == id_questão e data = data de hoje e tipo = "P"
        tentativa = 11
        // força se mantém
    fimSe
    Se (tentativa == 10)
        gravar no arquivo questoes_respondidas_[codigo do game].json no
        registro id_questao == id_questão e data = data de hoje e tipo = "E"
        tentativa = 21
        // força se mantém
    fimse
senão // resposta incorreta
    Se (tentativa == 00)
        gravar no arquivo questoes_respondidas_[codigo do game].json no
        registro id_questao == id_questão e data = data de hoje e tipo = "P"
        tentativa = 10
        força = força – 9
        Exibir pop up Perguntando se deseja continuar novamente
    senão
        Se (tentativa == 10)
            gravar no arquivo questoes_respondidas_[codigo do game].json
            no registro id_questao == id_questão e data = data de hoje e tipo =
            "P"
            tentativa = 20
            força = força – 9 // gravar no arquivo json
        fimse
    fimse
fim se
fimWhile

```

7.4. Fluxos Alternativos

A1. Retornar ao menu principal se não existe possibilidade de rega da planta

7.5. Fluxos de Exceção

7.6. Pré-condições

7.7. Pós-condições

7.8. Casos de Teste

7.9. Protótipo de Tela

Não existe tela.

8. Exibir Questão

8.1. Objetivo

Apresentar na tela a questão indicada e corrigir após resposta.

8.2. Ator

Aluno.

8.3. Fluxo Principal

P1. Entrada: id_questao – Acessar arquivo **questoes_[codigo do game].json** e obter:
enunciado

resp_correta

comentario

```
"respostas": [  
  {  
    "id_resp": "01",  
    "texto": "resposta 01"  
  },  
  {  
    "id_resp": "02",  
    "texto": "resposta 02"  
  },  
  {  
    "id_resp": "03",  
    "texto": "resposta 03"  
  },  
  {  
    "id_resp": "04",  
    "texto": "resposta 04"  
  }  
]
```

P2. Exibir tela com o texto da questão e as alternativas (respostas)

P3. Leitura da resposta do aluno

P4. Exibir pop up com indicação de “Correta” ou “Erro” para a resposta.

P5. Retornar para a função chamadora o ERRO (0) ou ACERTO (1)

8.4. Fluxos Alternativos

8.5. Fluxos de Exceção

8.6. Pré-condições

Caso de uso ativado sempre que o usuário seleciona um jogo.

Pode ser construído como uma função que recebe o código do jogo como parâmetro de entrada.

8.7. Pós-condições

8.8. Casos de Teste

Verificar se força atualizada conforme data do sistema.

8.9. Protótipo de Tela

Não existe tela.

9. Atualizar Força da Planta

9.1. Objetivo

Atualizar a força da planta.

9.2. Ator

Sistema (ativado sempre que o aluno selecionar um game).

9.3. Fluxo Principal

P1. Obter código do jogo.

P2. Obter força e data_atualização referente ao código do jogo obtido em P1.

Se data_atualização < data do sistema (data de hoje) então

diferença = data do sistema – data_atualização

força = força – 18 * diferença

data_atualização = data do sistema (data de hoje)

atualizar (gravar no arquivo games.json): campos “força” e “data_atualização”

fimSe

9.4. Fluxos Alternativos

9.5. Fluxos de Exceção

9.6. Pré-condições

Caso de uso ativado sempre que o usuário seleciona um jogo.

Pode ser construído como uma função que recebe o código do jogo como parâmetro de entrada.

9.7. Pós-condições

9.8. Casos de Teste

Verificar se força atualizada conforme data do sistema.

9.9. Protótipo de Tela

Não existe tela.

10. Utilizar Estrelinhas

10.1. Objetivo

Selecionar e estrelas para uso e atualizar a força da planta.

10.2. Ator

Aluno

10.3. Fluxo Principal

P1. Obter código do jogo.

P2. Obter “força” e “qtd_estrelinhas” referente ao código do jogo obtido em P1.

Força = força + qtd_estrelinhas

Se força > 100 então

qtd_estrelinhas = força - 100

força = 100

senão

qtd_estrelinhas = 0

P3. Atualizar (gravar no arquivo gamesdata.json): campos “força” e “qtd_estrelinhas”

10.4. Fluxos Alternativos

10.5. Fluxos de Exceção

10.6. Pré-condições

Caso de uso ativado pelo aluno.

10.7. Pós-condições

10.8. Casos de Teste

Verificar se força e quantidade de estrelas foram atualizadas.

10.9. Protótipo de Tela

Não existe tela.

11. Responder Pergunta

11.1. Objetivo

Apresentar pergunta ao aluno e verificar resposta.

11.2. Ator

Sistema (ativado por Obter Estrelinhas ou Regar Planta).

11.3. Fluxo Principal

Entrada: id_questao, tentativa

saida: tentativa, n {pontuação obtida na questão}

Obter enunciado da questão (Carregar pergunta do arquivo questoes_[codigoGame].json{procurar pela questão que possui o mesmo id_questao})

Se tentativa == 10 então

exibir "dica" ao aluno

fimSE

Exibir tela com a pergunta e as opções existentes.

Leitura da resposta do aluno: "O aluno seleciona uma opção e pressiona o botão para verificar se resposta correta"

se resposta correta então

switch tentativa:

"00": resp = 11

n=18

"10": resp = 21

n=9

fim switch

senão

n=0

switch tentativa:

"00": resp = 10

"10": resp = 20

fim switch

fim se

tentativa = resp

retornar ao chamador tentativa, n

P4. Retorna ao menu principal.

11.4. Fluxos Alternativos

Não se aplica.

11.5. Fluxos de Exceção

11.6. Pré-condições

Alterar para próxima pergunta se foram realizadas duas tentativas na pergunta anterior.

11.7. Pós-condições

11.8. Casos de Teste

Verificar retorno da função. Valores gravados corretamente no arquivo json.

11.9. Protótipo de Tela

figura
regador

[texto pergunta]

Tentativa: 1 ou 2

☐ resposta 1
☐ resposta 2
☐ resposta 3
☐ resposta 4

Texto de "dica"

OK Retornar

12. Obter Estrelas

12.1. Objetivo

Responder perguntas para acumular estrelinhas.

12.2. Ator

Aluno

12.3. Fluxo Principal

Valores para tentativa {bits tentativa-resultado}:

- 10: primeira tentativa resposta errada
- 11: primeira tentativa resposta correta
- 20: segunda tentativa resposta errada
- 21: segunda tentativa resposta correta

P1. Abrir arquivo **gamesdata.json** (relativa ao jogo selecionado pelo aluno) e obter:

1. Qtd_Estrelinhas

No arquivo: **questoes_respondidas_[codigo do game].json** - carregar vetor com dados das questões respondidas [data, id_questao, tentativa] com tipo == "E" {"E" para estrelinhas}

Contar a quantidade de questões respondidas na data de hoje (data do sistema)==>**qtd_resp**

Continuar = verdadeiro

Enquanto continuar e **qtd_resp<=9** faça

Verificar se existem questões com **data** == data de hoje (data do sistema) e (tentativa == 10 ou tentativa == 00)

{se existe utilize o primeiro registro encontrado}

Se não existe então

buscar questão no arquivo de questões: **questoes_disponiveis_[codigo do game].json**

selecione o primeiro registro (armazenar valor de *id_questão*) – remover o valor do arquivo **questoes_disponiveis_[codigo do game].json**

criar registro no arquivo **questoes_respondidas_[codigo do game].json** com:

data = data de hoje

id_questao = *id_questão*

tentativa = 00 // estado inicial

tipo = "E"

qtd_resp = qtd_resp+1

Ativar Caso de USO **Exibir Questão** (id_questao)

Se resposta correta então

gravar no arquivo **questoes_respondidas_[codigo do game].json** no registro **id_questao == *id_questão*** e **data = data de hoje** e **tipo = "E"**

tentativa = 11

atualizar estrelinhas: arquivo **gamesdata.json** localizar em regitro "**codigo**" == **codigo do game**: qtd_estrelinhas = qtd_estrelinhas + 2

senão // primeira resposta incorreta

gravar no arquivo **questoes_respondidas_[codigo do game].json** no registro **id_questao == *id_questão*** e **data = data de hoje** e **tipo = "E"**

```

        tentativa = 10
    fimSE
fim se
Se existe questao com data de hoje então // e tentativa == 10 ou tentativa == 00
    Ativar Caso de USO Exibir Questão (id_questao)
    Se resposta correta então
        Se (tentativa == 00)
            gravar no arquivo questoes_respondidas_[codigo do game].json no
            registro id_questao == id_questão e data = data de hoje e tipo = "E"
            tentativa = 11
            atualizar estrelinhas: arquivo gamesdata.json localizar em regitro "codigo"
            == codigo do game: qtd_estrelinhas = qtd_estrelinhas + 2
        fimSe
        Se (tentativa == 10)
            gravar no arquivo questoes_respondidas_[codigo do game].json no
            registro id_questao == id_questão e data = data de hoje e tipo = "E"
            tentativa = 21
            atualizar estrelinhas: arquivo gamesdata.json localizar em regitro "codigo"
            == codigo do game: qtd_estrelinhas = qtd_estrelinhas + 1
        fimse
    senão // resposta incorreta
        Se (tentativa == 00)
            gravar no arquivo questoes_respondidas_[codigo do game].json no
            registro id_questao == id_questão e data = data de hoje e tipo = "E"
            tentativa = 10
        senão
            Se (tentativa == 10)
                gravar no arquivo questoes_respondidas_[codigo do game].json
                no registro id_questao == id_questão e data = data de hoje e tipo =
                "E"
                tentativa = 20
            fimse
        fimse
    fim se

    Exibir pop up Perguntando se deseja continuar novamente
    // continuar recebe falso se deseja parar

```

fimWhile

P2. Retornar ao menu principal

12.4. Fluxos Alternativos

A1. Quantidade máxima de tentativas em responder perguntas foi atingida (são nove perguntas que podem ser respondidas por dia). ==> exibir mensagem informando que quantidade máxima de tentativas foi realizada.

12.5. Fluxos de Exceção

12.6. Pré-condições

Dados do game carregados quando aluno selecionou o game atual.

12.7. Pós-condições

Atualizar arquivo gamesdata.txt com informações sobre:

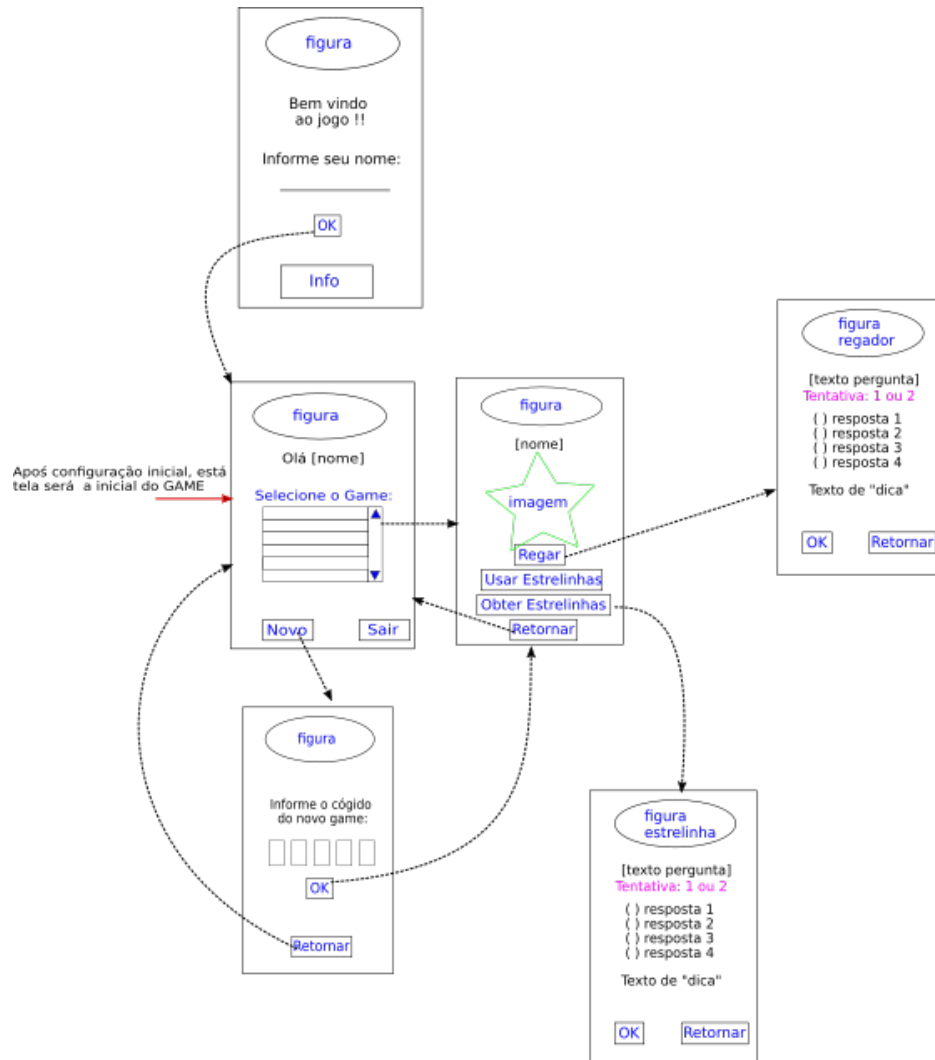
1. Qtd Estrelinhas: nova quantidade de estrelas acumuladas

12.8. Casos de Teste

12.9. Protótipo de Tela

Não há tela.

1. Transição de Telas



Detalhes de implementação

Para cada game criar um arquivo XML (**games.txt**), onde serão registrados:

1. Código do Game
2. Nome Fantasia: nome curto para o game
3. Disciplina:
4. Responsável:
5. Data início e fim;
6. Força: força atual da planta [inicial 100]
7. Data última rega:
8. Tentativa regar: indica qual tentativa o aluno está realizando na questão atual
9. Qtd Estrelinhas: [inicial 0 – zero]
10. Data obter estrelas: data da última atividade realizada para obter estrelas
11. Tentativa obter estrelas: indica a tentativa em responder a pergunta atual.
12. Perguntas_respondidas: perguntas respondidas na última de obtenção de estrelas [existe um limite]

O aplicativo apresenta os seguintes elementos:

1) Setup inicial, ou seja, criar o arquivo com os valores iniciais, como mostrado acima. [precisamos pensar em esconder o setup para que seja feito somente uma vez, ou desabilitar essa opção por alguns dias...]

2) Entrada de evento:

- Regar: para isso é necessário responder a pergunta [as perguntas serão incluídas em em outra fase do projeto]
- Usar Estrelinha: mostre a quantidade de estrelas disponíveis e faça a leitura da quantidade que deseja usar, atualize a a quantidade de estrelas e a força no arquivo;
- Obter estrelas: responder pergunta para aumentar o saldo de estrelas, será exibida uma pergunta, ao acertar a resposta, será creditado um certo valor ao saldo de estrelas do aluno.

3) Visualizador: exibe o pé de feijão,

Abrir arquivo binário (game.bin), obter a força e a quantidade de datas na variável **atividades**, exibir a imagem equivalente.

*** **Antes de cada ação**, seja de visualização ou inserção de evento (qualquer um dos listados no item 2), verifique a data da última inserção de dados no vetor atividades, se a última data é maior que 24 horas, inserir os registros necessários no vetor e atualize a força, nesse caso o aluno esqueceu de regar nesses dias.

- Plantar Feijão é o setup inicial.
- Visitar pé de feijão: visualizador, dependemos das imagens, use alguma imagem da internet (imagens diferentes) para teste;
- Regar: são duas fases:
 - Pergunta e resposta: carregar pergunta e resposta de um arquivo texto (nesse momento, deixar fixo no código uma pergunta com 4 opções de resposta), lembre-se que o aluno pode fazer duas tentativas, use um botão Verificar que processa a resposta e indica se está certa ou errada.
 - determinar a quantidade de água de acordo com a resposta (use o trecho de código apresentado).
- Usar Estrelinha: exibir a quantidade atual (obter do arquivo binário), leitura da quantidade desejada para uso, atualizar a força no arquivo e a quantidade de estrelinhas restantes.
- Obter Estrelinha:
 - pergunta e resposta: carregar pergunta, resposta e a quantidade de estrelinhas que vale a questão, carregar do arquivo texto, o aluno tem direito a somente uma tentativa. Em caso de acerto, atualizar a quantidade de estrelinhas no arquivo binário.

Sobre o app, em linhas gerais, penso em criar um serviço (site) para o professor, os passos seriam os seguintes:

- a) a professora realiza seu cadastro, depois seleciona a disciplina e conteúdo que gostaria de aplicar a seus alunos, prazos, etc (ou seja, inicia uma atividade/game). O site gera um código para essa atividade.
- b) a professora repassa o código aos alunos que vão participar da atividade;
- c) cada um dos alunos instala o app e realiza a configuração inicial (insere seu nome).
- d) após a configuração inicial o aluno insere o código da atividade (fornecido pelo prof).
- e) o app carrega as perguntas referentes a essa atividade (referente à disciplina/conteúdo definidos pelo prof).
- f) ao concluir sua atividade o app envia os resultados ao site (definir quais informações são relevantes ao prof)

g) o site reúne essas informações e envia estatística ao prof (podem ser gerais da turma ou por aluno).