
Avaliação referente ao RA1

ATENÇÃO: PJBL é aprendizagem Baseada em Projeto e refere-se ao desenvolvimento da solução de um projeto **EM EQUIPES**. Logo, esta atividade **NÃO PODERÁ SER desenvolvida de maneira individual**.

A Professora liberará o cadastro das equipes durante o horário da aula de divulgação dessa atividade. **A responsabilidade de cadastramento da equipe é de cada estudante** e a equipe deverá ser composta por **3 (três) integrantes** da mesma turma (das aulas de implementação).

Objetivo geral:

Desenvolver um aplicativo orientado a objetos em **Python**, sua interface gráfica em **Kivy** e respectivo Banco de Dados em **SQLite** que satisfaçam os diagramas e solicitações descritas abaixo.

Sobre as equipes e os prazos:

1. Equipes: com **3** estudantes. Conforme a quantidade de estudantes nas turmas, poderá existir, no máximo, 1 (UMA) dupla se for necessário.
2. Itens de entrega no Blackboard:
 - a. **arquivo (py)** com o código fonte orientado a objetos em Python, bem-organizado e devidamente documentado (com comentários para facilitar a compreensão)
 - b. **arquivo (kv)** com a interface gráfica em Kivy e referente ao programa Python entregue. Os componentes gráficos **OBRIGATÓRIOS** a serem utilizados são:
Labels, Spinner, Buttons, TextInputs e Layouts
 - c. **arquivo (db)** com a base de dados em SQLite 3, com todas as tabelas devidamente **populadas**, ou seja, **cada tabela deverá ter, no mínimo, 10 (dez) registros cadastrados**. As tabelas deverão respeitar o Diagrama Entidade Relacionamento quanto às suas chaves (primárias e estrangeiras)
 - d. **arquivo (pdf)** com nome completo dos membros da equipe, descrição resumida (um parágrafo de, no máximo, 10 linhas) do aplicativo e **instruções e exemplos de uso do programa**
 - e. **um vídeo de, no máximo, 4 minutos** mostrando o funcionamento completo do aplicativo (**não é preciso explicar o código, basta mostrar o programa funcionando**) - no vídeo, **todos os integrantes da equipe devem participar!**
Criem um roteiro de apresentação do funcionamento completo do seu aplicativo, simulando uma partida e apresentando os resultados obtidos. LEMBREM-SE: todos os integrantes da equipe deverão ser identificados nas imagens do vídeo.
Esse vídeo deverá ser postado pela equipe no **YOUTUBE** com **acesso restrito** e seu link deverá estar colocado no campo ADICIONAR COMENTÁRIOS da entrega da atividade no BlackBoard, conforme Figura-1 de exemplo.

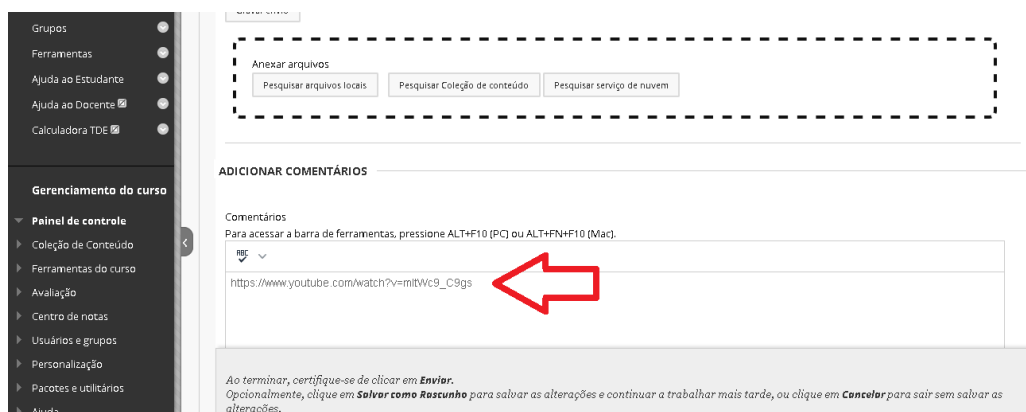


Figura 1. Exemplo de inserção do link do vídeo.

3. Avaliação:

- itens e link entregues no Blackboard até **20/06 (domingo)** para receber NOTA INTEGRAL
- entregas atrasadas serão aceitas, no máximo, até 22/06 com os devidos descontos computados
- defesa do projeto **SOMENTE no horário de aula na semana de 21/06 a 25/06**

4. Período para desenvolvimento do projeto pelas equipes:

a partir de 07/06 até a data da entrega do projeto, todos os estudantes poderão usar todos os horários de aula (CSBA - turma completa) para reunirem-se com suas equipes e focarem no desenvolvimento da sua solução

Sobre os descontos previstos:

ATENÇÃO às penalidades a serem aplicadas para entregas indevidas:

Descrição dos problemas considerados:	Desconto
Entrega feita em local indevido (diferente do INDICADO no BlackBoard)	2 pontos
Entrega feita após data limite: 20/06 às 23h59	¹ 2 pontos por dia de atraso
Plágio entre resultados entregues por equipes distintas	10 pontos
Arquivo(s) entregue(s) pela equipe diferentes do conteúdo solicitado	10 pontos
O resultado entregue pela equipe não segue o solicitado (formatos dos arquivos a serem entregues conforme descrito no item (2) Sobre as equipes e os prazos)	5 pontos
O aplicativo entregue pela equipe não é executado com sucesso (não roda - apresenta algum erro de sintaxe)	5 pontos

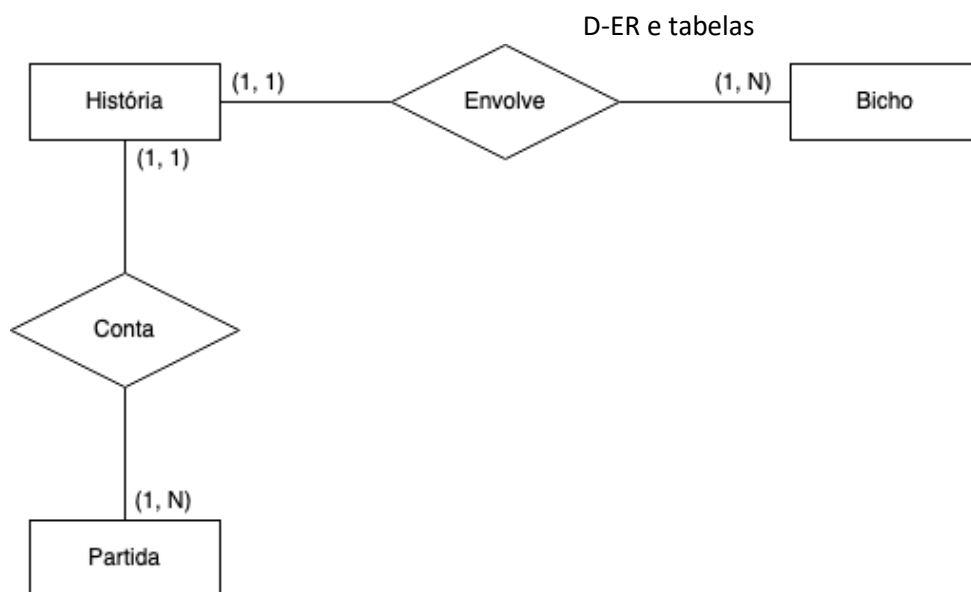
- Se houver descontos, estes serão computados inicialmente na correção. Na sequência, a correção seguirá rubrica disponibilizada no BlackBoard, considerando a nota máxima **com desconto**.
- Casos especiais serão tratados individualmente.

Sobre os dados de entrada:

¹ Nesse caso o desconto refere-se a 2 (dois) pontos a menos **por DIA de atraso**, NÃO considerando horas de atraso. Assim: entrega atrasada feita no dia seguinte **independe** da quantidade de horas de atraso dessa entrega.

1. O aplicativo deverá interagir com as tabelas do banco de dados, conforme especificado
2. O conjunto de dados deverá ter, ao menos, 10 registros cadastrados por tabela, entre números e textos, com significado para a aplicação e respeitando as cardinalidades especificadas entre elas
3. Cada equipe deverá cadastrar o seu próprio conjunto de dados a ser utilizado junto ao aplicativo
4. Caso alguma equipe queira acrescentar uma tabela ou campos junto aos dados, deve, primeiramente, confirmar sua validade com a respectiva Professora

Sobre a representação a ser seguida na implementação dos dados:



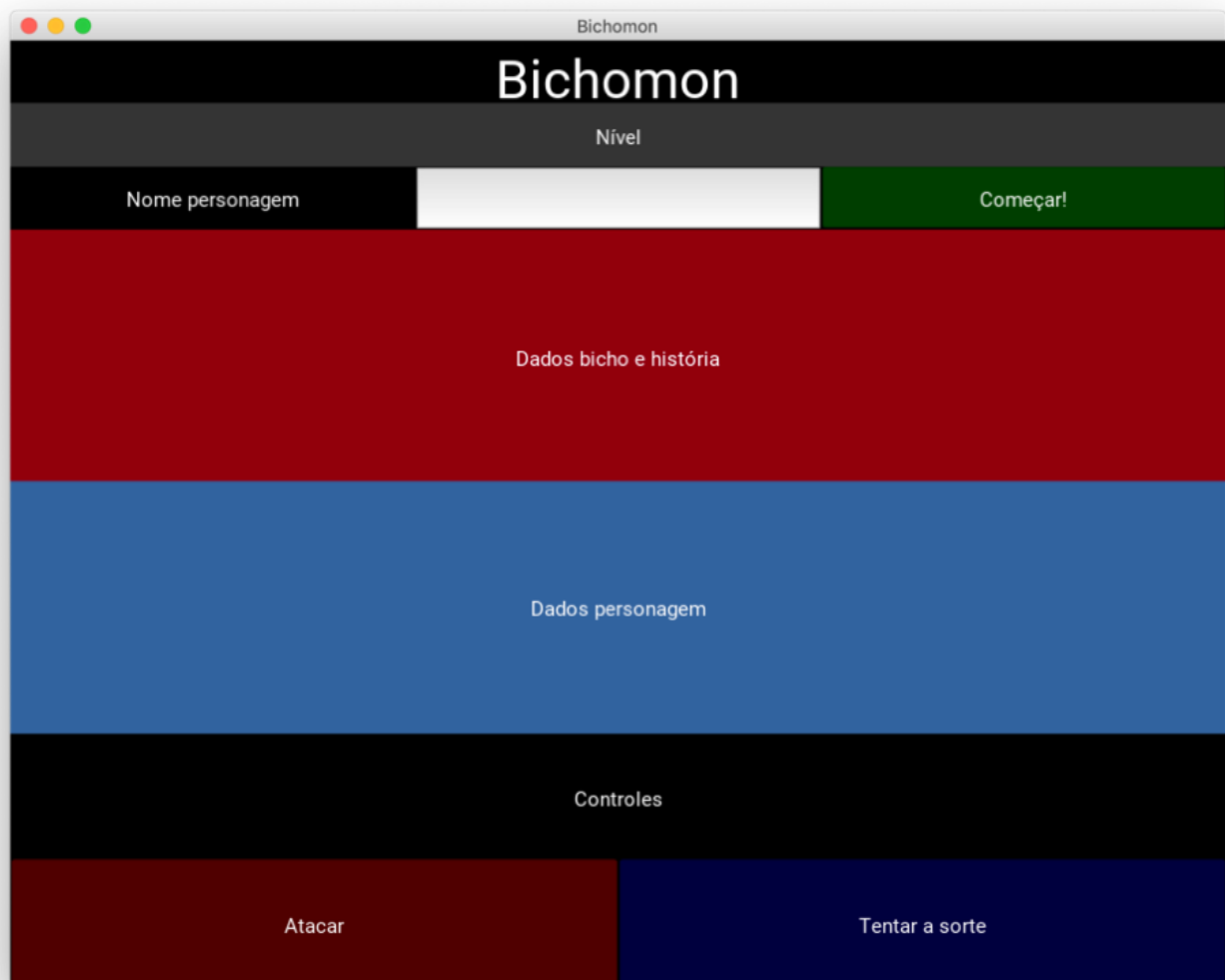
Historia	
PK	<u>codHistoria</u>
	textoIntro
	tipo

Bicho	
PK	<u>codBicho</u>
FK	codHistoria
	nome
	pontosEnergia
	ataqueMax
	ataqueMin

Partida	
PK	<u>codPartida</u>
FK	codHistoria
	nomePersonagem

Sobre a GUI - interface gráfica com o usuário em Kivy:

1. Elementos gráficos a serem usados:
No mínimo: Label, Spinner, Botões, Text input e Layouts
Caso a equipe queira utilizar mais alguns elementos gráficos distintos dos citados aqui, deve, primeiramente, confirmar sua validade com a respectiva Professora
2. Interface gráfica proposta:



Sobre a codificação dos programas:

1. O programa deve ser escrito em linguagem Python e com o paradigma orientado a objetos, importando todas as bibliotecas necessárias e disponibilizadas pelo próprio Python, pelo Kivy e integração com SQLite3.
2. O programa deve ser rigorosamente estruturado em classes e seus respectivos atributos e métodos, respeitando argumentos e valores de retorno necessários para o correto funcionamento do aplicativo como um todo
3. Fazer uso devido das regras de sintaxe definidas para a integração dos programas Python e Kivy, suas classes, eventos e métodos, variáveis comuns, argumentos
4. Devem ser usados NOMES SIGNIFICATIVOS para variáveis, métodos, classes e argumentos
5. Devem ser inseridos comentários no código de forma a facilitar a sua compreensão.
6. Deve ser feita a necessária e suficiente verificação da codificação dos programas, integração entre eles, verificação de dados válidos nas tabelas da base de dados, a fim de evitar erros de execução devido a dados inválidos ou ausentes.

Sobre a lógica a ser desenvolvida:

- **Elaboração de um jogo de batalha. Os dados das criaturas e da história vêm do banco dedados.**
- **Funcionamento geral:**
 1. Escolher o tipo de história (fácil ou difícil)
 2. Informar o nome do personagem
 3. Pressionar o botão 'começar'
 - a. Ao pressionar o botão começar, o será feita uma consulta no banco por todos os monstrosque são do mesmo tipo de história escolhida.
 - b. Criação da instância da classe Personagem com os valores padrões de inicialização
 - c. Exibir o texto de introdução da história
 - d. Exibir os dados do personagem na caixa correspondente
 4. Comandos
 - A. Atacar
 - Chama o método de atacar do personagem
 1. Sorteia um valor entre o ataque mínimo e máximo do personagem
 2. Chama o método para executar todo o turno da partida
 - B. Tentar a sorte
 - Chama o método de ataque crítico do personagem
 1. Sorteia um valor entre 0 e 10. Se for menor que 2, então chama a função de dardano no monstro passando 100 como parâmetro. Senão, multiplica o valor sorteado por 3 e acrescenta o resultado no total de pontos de energia do personagem
 2. Chama o método para executar todo o turno da partida

5. Estrutura do banco:

- a. História (codHistoria, textoIntro, tipo)
- b. Bichos (codBicho, nome, pontos de energia, pontos de ataque mínimo, pontos de ataque máximo, codHistoria)
- c. Partida (codPartida, codHistoria, nomePersonagem)

Detalhamento dos métodos (Consultar diagrama de classes e de objetos presentes nesse documento)

1. Classe Personagem

a. Atacar ()

- Objetivo: calcular o valor do ataque do personagem
- Retorna o número sorteado que está entre os valores de ataque mínimo e máximo do personagem

b. Crítico ()

- Objetivo: calcular o valor do ataque do personagem
- Sorteia um número entre 0 e 10
- Se o número sorteado for menor que 2, retorna 100 para valor do ataque

Senão, retorna 0 para o valor do ataque do personagem e utiliza o número sorteado para definir uma taxa de recuperação de pontos de energia do personagem (máximo 50 pontos)

c. SofreDano (Inteiro: dano)

- Objetivo: diminuir pontos de energia do personagem
- Retira dos pontos de energia do personagem o valor passado como parâmetro no método (dano)

2. Classe Bicho

a. Atacar ()

- Objetivo: calcular o valor do ataque do personagem
- Retorna o número sorteado que está entre os valores de ataque mínimo e máximo do bicho

b. SofreDano (Inteiro: dano)

- Objetivo: diminuir pontos de energia do bicho
- Retira dos pontos de energia do personagem o valor passado como parâmetro no método (dano)

3. Classe História

a. ExibeIntro ()

- Objetivo: retornar o valor do texto da introdução
- Retorna o valor do atributo 'textoIntro', que armazena o texto da introdução recebida pelo banco de dados

4. Classe Partida

a. ExecutaTurnoPartida (Lógico: crítico)

- Objetivo: Realizar todas as ações que envolvem um turno na partida (ataque do personagem; ataque do bicho; verificação de situação dos pontos de energia do personagem e do bicho; verificação de situação da partida)
- Verifica o tipo de ataque do personagem (crítico ou comum)
- Executa o dano no bicho
- Atualiza os pontos de energia do bicho
 - Se ele foi derrotado, verifica se a jornada terminou ou se chama o próximo bicho
- Se o bicho não foi derrotado, executa dano no personagem
- Verifica se o personagem ainda tem pontos de energia
 - Se ele foi derrotado, finaliza o jogo. Senão, continua

5. Classe UtilizaBanco

a. CadastraPartida (Partida: partida)

- Objetivo: Cadastrar os dados da partida no banco de dados
- Acessar o banco de dados

b. Cadastrar os dados da partida utilizando as informações recebidas por parâmetro (partida)

c. RecuperaDadosHistoria (Partida: partida)

- Objetivo: Acessar e recuperar as informações sobre o tipo de história selecionada
- Acessar o banco de dados
- Recuperar os dados da história cadastradas de acordo com o tipo da partida escolhido. O tipo de história está armazenado na variável recebida como parâmetro
- Retornar os dados da história
 - Caso existam mais de uma história do mesmo tipo (nível de dificuldade selecionado), sorteia qual delas será retornado

d. RecuperaListaBichos (Partida: partida)

- Objetivo: Acessar e recuperar as informações de todos os tipos ligados à história selecionada
- Acessar o banco de dados
- Recuperar os dados de todos os bichos de acordo com o identificador da história. O código da história está armazenado na variável recebida como parâmetro
- Retornar a lista de bichos

Sobre o modelo a ser seguido na implementação dos programas:

Diagrama de Classes e de Objetos:

