COLÉGIO CAESP
Educação Infantil, Ensino Fundamental e Médio - EJA Fundamental e Médio
Rua Almirante Barroso, 1086 - Centro
Foz do Iguaçu - PR - CEP 85851 010 - Fone (45) 3523 2887
e-mail: caesp@caesp.net - www.caesp.net

LÓGICA DE PROGRAMAÇÃO E INTELIGÊNCIA ARTIFICIAL

Prof. Sara Torres

MATERIAL DE REVISÃO – 4º BIMESTRE

8º ANO

1 INTRODUÇÃO

Este material reúne os conceitos exigidos na avaliação: construção e revisão de prompts, criação de aplicativos com ferramentas de desenvolvimento, redes neurais artificiais e lógica de jogos no Scratch. O objetivo é compreender os conceitos, identificar erros e aplicar o conhecimento em situações práticas, e não apenas memorizar respostas.

2 O QUE É UM PROMPT?

Prompt é a instrução enviada a um sistema de inteligência artificial (IA). Pode ser uma pergunta, uma solicitação de texto ou uma descrição detalhada de um aplicativo. A qualidade da resposta depende, entre outros fatores, da clareza das instruções, das informações disponíveis e das limitações da ferramenta. Um prompt bem escrito melhora a probabilidade de obter um resultado útil, mas não garante que ele esteja correto.

2.1 Os sete elementos de um bom prompt

Objetivo: Dizer exatamente o que deve ser produzido: um aplicativo, uma explicação, um jogo ou uma tabela.

Contexto: Explicar o problema que se pretende resolver e onde o resultado será utilizado.

Público-alvo: Informar quem utilizará o produto: estudantes do 8º ano, professores, famílias etc.

Funcionalidades ou conteúdo: Especificar os recursos obrigatórios e o comportamento esperado.

Formato e interface: Definir como apresentar o resultado: telas, botões, linguagem, cores, campos e organização.

Restrições realistas: Indicar limites técnicos e condições de uso, evitando solicitações impossíveis ou contraditórias.

Critérios de verificação: Explicar como saber se o resultado funciona, incluindo exemplos de testes.

2.2 Como construir e aperfeiçoar um prompt

Primeiro, identifique a necessidade. Depois, transforme expressões vagas em ações observáveis. Acrescente dados de entrada, operações, resultados esperados e critérios de teste. Execute o prompt, observe o resultado e refine apenas o que ficou incompleto ou incorreto. Essa prática é chamada de iteração: testar, avaliar e reformular.

Exemplo ruim – aplicativo financeiro

“Faça um app de dinheiro que seja legal.”

Problema: “legal” é subjetivo; não há definição de público, campos, cálculos ou funcionalidades.

Exemplo melhor – aplicativo financeiro

“Crie um aplicativo simples de controle financeiro pessoal para adolescentes. Inclua: (1) cadastro de receitas com valor, data e descrição; (2) cadastro de despesas com categoria; (3) cálculo automático do saldo = total de receitas − total de despesas; (4) lista de movimentações com opção de editar e excluir; (5) filtro por mês. Use interface

em português, botões claros e layout adaptado ao celular. Não permita valores negativos nos campos de valor. Teste o saldo após adicionar, editar e excluir lançamentos.”

O segundo exemplo apresenta mais de quatro funcionalidades, além de público-alvo, interface e testes. Uma resposta gerada por IA ainda precisa ser conferida.

2.3 Instruções contraditórias e ambíguas

Uma contradição acontece quando duas exigências não podem ser cumpridas simultaneamente nas condições descritas. Por exemplo: exigir funcionamento totalmente sem internet e, ao mesmo tempo, consulta obrigatória a um servidor on-line em tempo real mesmo sem conexão. Sem uma conexão disponível, o aplicativo não consegue buscar novas informações do servidor.

Correção possível

“Crie um aplicativo que funcione sem internet usando dados salvos no aparelho. Quando houver conexão, sincronize os dados com o servidor e mostre a data da última atualização. Informe ao usuário quando as informações estiverem desatualizadas.”

Ambiguidade é a falta de precisão. “Faça um aplicativo bonito e fácil” não esclarece quais telas, tarefas ou padrões de acessibilidade são necessários. Uma instrução detalhada não precisa ser longa sem motivo: deve ser específica e coerente.

2.4 Outros exemplos de prompts bons e ruins

Ruim – explicação de IA

“Explique redes neurais.”

Melhor – explicação de IA

“Explique redes neurais artificiais para estudantes do 8º ano, em linguagem acessível, com um exemplo de reconhecimento de animais e a função das camadas de entrada, ocultas e de saída. Termine com duas perguntas de revisão.”

Ruim – aplicativo de estudos

“Crie um aplicativo de estudos.”

Melhor – aplicativo de estudos

“Crie um aplicativo de organização de estudos para alunos do 8º ano. Inclua cadastro de disciplinas e tarefas com prazos, calendário semanal, marcação de atividades concluídas e lembretes visuais. Apresente uma tela inicial com tarefas pendentes, menu simples, letras legíveis e interface em português. Teste a inclusão e a conclusão de tarefas.”

3 REPLIT E BASE44: FERRAMENTAS PARA CRIAR PROJETOS

3.1 Replit

Replit é um ambiente de desenvolvimento acessado pelo navegador que permite escrever, executar, testar e organizar código. Pode ser utilizado em projetos com linguagens como Python, HTML, CSS e JavaScript, conforme os recursos disponíveis. Em projetos web, HTML organiza o conteúdo, CSS define a aparência e JavaScript acrescenta interações. Recursos assistidos por IA podem ajudar a gerar e modificar código, mas o desenvolvedor precisa revisar o resultado.

Exemplo de uso: desenvolver uma página com formulário de despesas e um botão para calcular o saldo. Para corrigir um erro, o estudante pode examinar mensagens de execução, verificar nomes de variáveis e testar cada operação separadamente.

3.2 Base44

Base44 é uma plataforma de criação de aplicativos com apoio de IA, na qual o usuário descreve em linguagem natural o que deseja construir. A plataforma pode gerar interfaces e comportamentos a partir da descrição e permitir ajustes posteriores. O resultado depende do pedido, dos recursos disponíveis e das configurações do projeto; funcionalidades como armazenamento, autenticação e integrações devem ser verificadas na aplicação criada.

Exemplo de uso: pedir um aplicativo de estudos com calendário, cadastro de tarefas,

disciplinas e visualização de pendências. Após a geração, verificar se os botões respondem, se as informações são mantidas como esperado e se a interface é compreensível.

3.3 Comparação e cuidados

Replit costuma oferecer maior contato direto com o código e com a execução do programa; Base44 privilegia a descrição do aplicativo por meio de prompts e ajustes orientados por IA. Nenhuma ferramenta elimina a necessidade de planejar, testar e corrigir. Antes de publicar um aplicativo, evite expor senhas, dados pessoais ou informações sensíveis.

4 COMO IDENTIFICAR E CORRIGIR ERROS

Erros podem surgir tanto na formulação do prompt quanto na execução do programa. O procedimento recomendado é reproduzir o problema, identificar a causa provável, modificar uma parte de cada vez e testar novamente.

Pedido vago: Sintoma: faltam funções. Correção: indicar campos, botões, regras e exemplos de uso.

Instruções incompatíveis: Sintoma: o sistema ignora uma exigência. Correção: escolher prioridades e definir comportamento com e sem conexão.

Botão sem ação: Sintoma: clicar não altera a tela. Correção: verificar evento de clique e função associada.

Saldo incorreto: Sintoma: despesas são somadas como receitas. Correção: conferir fórmula, sinal dos valores e testes com números conhecidos.

Dados não persistem: Sintoma: registros desaparecem ao atualizar a página. Correção: verificar se existe mecanismo de armazenamento e se ele está configurado.

Variável com nome incorreto: Sintoma: erro de execução ou valor indefinido. Correção: conferir a escrita exata e a inicialização da variável.

Condição lógica errada: Sintoma: perde vida ao tocar em objeto que deveria dar pontos. Correção: revisar o teste condicional e o objeto detectado.

Treinamento insuficiente: Sintoma: IA erra imagens novas. Correção: revisar variedade e rotulagem dos dados e avaliar com imagens não vistas.

4.1 Exemplos de comandos de correção

Correção de saldo

“O aplicativo calcula o saldo incorretamente. Corrija para saldo = receitas − despesas. Não altere o restante da interface. Teste com receitas de R$ 100 e despesas de R$ 35; o saldo esperado é R$ 65.”

Correção de formulário

“Ao clicar em Salvar, a tarefa não aparece na lista. Verifique o evento do botão, a validação do campo de descrição e a atualização da lista. Explique a causa identificada e teste o cadastro de duas tarefas.”

Um bom pedido de correção descreve o comportamento observado, o comportamento esperado, o contexto do erro e um teste verificável. Mensagens como “não funciona, arrume tudo” dificultam o diagnóstico.

5 REDES NEURAIS ARTIFICIAIS

Redes neurais artificiais são modelos computacionais inspirados em alguns aspectos da organização de neurônios biológicos. Elas processam informações por meio de unidades matemáticas conectadas e ajustam parâmetros durante o treinamento para reconhecer padrões. Não funcionam exatamente como um cérebro humano e não se deve concluir que possuam consciência.

5.1 Camadas de uma rede neural

Camada de entrada: recebe os dados fornecidos ao modelo, como valores que representam pixels de uma imagem. Camadas ocultas: realizam transformações intermediárias e combinam características relevantes dos dados. Camada de saída: produz o resultado, como probabilidades atribuídas às classes “cão” e “gato”.

Fluxo simplificado: DADOS (imagem) → CAMADA DE ENTRADA → CAMADAS OCULTAS → CAMADA DE SAÍDA (classificação). A decisão final pode ser incorreta,

mesmo quando o sistema apresenta alta confiança.

5.2 Treinamento, validação e testes

No aprendizado supervisionado, o sistema recebe exemplos com respostas conhecidas, chamados de dados rotulados. Durante o treinamento, compara suas previsões aos rótulos e ajusta parâmetros para reduzir os erros. Um conjunto de validação pode orientar ajustes do modelo; um conjunto de teste separado permite avaliar seu desempenho em dados não utilizados no treinamento. O desempenho deve ser analisado em imagens variadas, inclusive com mudanças de iluminação, ângulo e fundo.

Se uma rede foi treinada principalmente com fotos de cães e gatos, uma imagem de cavalo pode ser classificada incorretamente. O modelo pode não ter aprendido características suficientes para reconhecer uma classe diferente; isso não significa que ele “decidiu errar”. Uma rede treinada apenas para distinguir cães e gatos pode nem possuir a opção “cavalo” na saída.

5.3 Estudo de caso: separação de resíduos recicláveis

Objetivo: identificar, por fotografia, materiais como papel, plástico, metal e vidro.

Etapa 1: coletar imagens variadas e obter rótulos corretos para cada categoria. Etapa 2: organizar dados de treinamento, validação e teste, evitando imagens duplicadas entre os conjuntos. Etapa 3: treinar o modelo para associar características visuais às categorias. Etapa 4: testar com novas imagens, diferentes fundos e objetos amassados ou parcialmente ocultos. Etapa 5: registrar os erros, comparar as categorias mais confundidas e melhorar os dados ou o modelo. Um sistema real também precisa lidar com materiais mistos, resíduos sujos e objetos que não pertençam às classes previstas.

Atenção: boa precisão em fotos de treinamento não comprova que o modelo funcionará bem em situações novas. Os testes servem justamente para avaliar sua capacidade de generalização.

6 SCRATCH E A LÓGICA DE JOGOS

Scratch é uma linguagem de programação visual baseada em blocos. Para criar um jogo em que um personagem se mova com as setas, colete objetos e perca vidas ao tocar em inimigos, é necessário combinar diferentes categorias de comandos.

Eventos: Iniciam ações, como “quando a bandeira verde for clicada” ou “quando a tecla for pressionada”.

Movimento: Alteram a posição do personagem no cenário.

Sensores: Detectam teclas pressionadas, contato com objetos e outras condições.

Controle e condicionais: Executam repetições e decisões, como “se tocar no inimigo, então...”.

Variáveis: Armazenam valores que mudam, como pontos, vidas e tempo.

Operadores: Permitem comparações e cálculos, por exemplo verificar se vidas = 0.

6.1 Exemplo de lógica de jogo

Ao clicar na bandeira verde: definir pontos = 0 e vidas = 3. Repetir continuamente: se a seta para a direita estiver pressionada, mover o personagem para a direita; se tocar em um objeto coletável, aumentar pontos em 1 e reposicionar o objeto; se tocar em um inimigo, diminuir vidas em 1 e impedir múltiplas perdas instantâneas pelo mesmo contato; se vidas = 0, mostrar “Fim de jogo” e interromper a partida.

Erros comuns: esquecer de inicializar as variáveis; não detectar o toque; repetir a perda de vidas a cada instante de contato; não reposicionar o item coletado; usar apenas blocos de aparência e som sem lógica de interação.

7 ATIVIDADES DE FIXAÇÃO

1. Por que pedir “um app bonito” não é suficiente para definir um projeto?

2. Escreva quatro funcionalidades essenciais de um aplicativo de controle

financeiro.

3. Como um aplicativo pode funcionar sem internet e atualizar dados quando a conexão voltar?

4. Qual é a diferença entre um prompt genérico e um prompt com público-alvo e formato definidos?

5. Explique uma diferença entre Replit e Base44.

6. Uma tarefa desaparece depois de atualizar a página. Qual aspecto do aplicativo deve ser investigado?

7. Diferencie as camadas de entrada, ocultas e de saída de uma rede neural. 8. Por que testar uma rede neural com imagens que não foram usadas no treinamento? 9. Descreva as etapas para treinar uma IA que reconheça papel, plástico, metal e vidro.

10. Quais blocos do Scratch ajudam a detectar inimigos e reduzir a variável vidas?

11. Reescreva: “Crie um app de estudos”, incluindo público-alvo, três funções e uma característica visual.

12. Um aplicativo deveria mostrar R$ 65, mas mostra R$ 135 após registrar R$ 100 de receita e R$ 35 de despesa. Identifique o erro e proponha um teste.

8 ORIENTAÇÕES FINAIS DE ESTUDO

Para estudar, procure explicar os conceitos com suas próprias palavras e praticar a reescrita de prompts. Em situações-problema, identifique o objetivo, os dados disponíveis, a operação necessária e como verificar o resultado. Lembre-se de que o desenvolvimento de software e o treinamento de IA exigem testes e correções sucessivas.

REFERÊNCIAS

SCRATCH FOUNDATION. Scratch. Disponível em: https://scratch.mit.edu/. Acesso em: 17 nov. 2026.

REPLIT. Replit. Disponível em: https://replit.com/. Acesso em: 17 nov. 2026.

BASE44. Base44. Disponível em: https://base44.com/. Acesso em: 17 nov.

2026.

GOOGLE. Machine Learning Crash Course. Disponível em: https://developers.google.com/machine-learning/crash-course. Acesso em: 17 nov. 2026.