TEXTO

Os jogos foram divididos por “temas” para que as estruturas semelhantes fossem desenvolvidas linearmente, otimizando o processo. Foram delimitados 5 temas sendo eles: interações com pessoas (perguntas e respostas para encontrar a pessoa correta ou dados sobre ela) (jogos 1 e 2), comidas (jogos 3 e 4), roupas (jogos 5 e 6), vocabulário de viagem (jogos 7 e 8) e vocabulário de casa (jogo 9). O jogo 10 foi pensado para ser um compilado de temas abordando festas típicas brasileiras. Para o front-end, ou seja, o desenvolvimento da interface da plataforma e dos jogos, foi escolhido o REACT, uma biblioteca JavaScript gratuita e de código aberto para a construção de interface de usuário e seus componentes. E para o back-end, onde as informações dos jogos e os dados gerados pelos usuários ficam armazenados, decidiu-se pelo Strapi, um sistema de gestão de conteúdo “sem cabeça”, que dizer, independente do front-end, que gera uma API para a administração dos dados.

**Estruturação do banco de dados**

O banco de dados foi pensado já considerando a ferramenta de criação e edição que será feita para os professores do CCBF. O que fizemos foi, ao invés de pensar em jogos fechados, eles são “tipos de jogos”, onde o professor vai poder criar mais atividades desse mesmo tipo, controlando a dificuldade e o conteúdo apresentado neles. Por exemplo, o jogo 1 tem por objetivo conversar com as pessoas presentes numa sala para coletar informações sobre elas e preencher os dados que faltam em sua lista de contatos. Os professores poderão criar um jogo desse mesmo tipo, mudando os personagens, quais perguntas podem ser feitas, quantidade limite de perguntar por sala, quais dados estão faltando na lista de contato, quantas salas/ambientes terão o jogo e assim por diante.

Assim, as tabelas do banco de dados foram construídas separadamente para cada tipo de jogo, além de tabelas que guardam dados que podem ser utilizados em mais de um tipo de jogo, como assets (que guarda a imagem, o nome do objeto e seu tipo), personagens e roupas. No banco de dados os jogos foram chamados de missões e cada missão guarda o tipo de jogo ao qual ela pertence. Além disso, a tabela base das missões guarda seu nome e descrição e, pensando na utilização dessa plataforma em outras embaixadas em diferentes países, criamos a possibilidade de adicionar várias traduções para ambos os textos, especificando a qual idioma a tradução pertence. Quando o aluno entrar no jogo, a tradução será mostrada de acordo com o idioma selecionado por ele ao criar seu perfil.

A tabela base das missões contém dados comuns entre todas elas, ou seja, além das informações descritas acima, temos música de fundo, ícone inicial e duas variáveis booleanas que determinam se o jogo ainda está na fase de montagem ou se ele já está pronto para ser usado pelo aluno e se os cliques do aluno durante o jogo devem ser armazenados. Para os dados específicos de cada missão foi criada uma tabela, por padrão chamada de *GameNumberMissionData* (*GameOneMissionData*, para o jogo 1, por exemplo), contendo dados e relações com outras tabelas para obter as informações necessárias para que o jogo funcione como deveria. A seguir serão descritas as tabelas para cada tipo de jogo e como os dados serão utilizados.

Como descrito anteriormente, no jogo 1 (Crise de meia idade) o aluno terá que conversar com personagens em uma sala para obter as informações que faltam na lista apresentada. Esse jogo foca em testar a capacidade de o aluno de utilizar cumprimentos adequados para iniciar uma conversa e identificar as perguntas necessárias para obter os dados que ele necessita. A tabela de dados da missão contém as perguntas que podem ser feitas, as salas (ou locais), textos de transição entre as salas e os textos de feedback. Aqui, a tabela de salas do jogo 1, chamada de *GameOneLocation*, guarda a relação com os personagens que aparecem, o número máximo de perguntas que podem ser feitas, a ordem na qual a sala aparecerá e a relação com a tabela geral de locais, a qual guarda uma imagem de fundo e o nome do local.

Os personagens, na tabela *GameOneMissionCharacter*, guardam quais informações (nome, país ou emprego) vão ou não ser mostradas, as repostas para cada uma das perguntas que podem ser feitas, como o personagem terminará o diálogo (que aparecerá se o aluno chegar ao número máximo de perguntas que ele pode fazer na sala) e a relação com a tabela geral de personagens, que possui os dados do personagem e suas imagens. A parte de *feedback*, ou seja, a mensagem que o aluno receberá ao terminar o jogo, o intervalo de pontuação referente a mensagem em questão, as imagens que aparecerão e as possíveis traduções para a mensagem. Por fim, para cada jogo temos uma tabela que guarda o resultado obtido pelo jogador. O jogo 1 guarda apenas os dados básicos de pontuação e se o jogador ganhou ou não o jogo.

O jogo 2 (Um novo começo) tem uma proposta parecida com o jogo anterior, porém seu objetivo agora é encontrar uma pessoa específica, seguindo as dicas obtidas em diálogos com os personagens. Por ser um jogo para o segundo nível do curso, ele busca avaliar não só as áreas do jogo anterior, mas também a capacidade do aluno de entender descrições de aparência física e condição emocional. Na tabela principal *GameTwoMissionData* armazenam-se as informações de locais (já que os personagens aqui também ficam divididos em salas), as perguntas que podem ser feitas nos diálogos, a lista de personagens do jogo e as informações do tutorial.

Diferente do jogo 1, onde o “custo” de fazer perguntas que não fazem parte do contexto é apenas gastar uma oportunidade dentro no número máximo de diálogos que podem ser formados, aqui, caso o jogador crie uma conversa que não faz sentido para o contexto, ele não conseguirá obter a dica que pode (ou não) estar presente na conversa do personagem. Assim, além da tabela de perguntas conter o texto da pergunta em si, ela terá uma variável booleana indicando se a pergunta é correta ou não. Neste jogo personagens não estão diretamente atrelados a uma sala específica assim, a variável de locais é ligada diretamente a tabela padrão da mesma e a divisão dos personagens entre elas é feita diretamente no código, sendo eles divididos aleatoriamente entre as salas cada vez que o jogo é iniciado.

Os personagens, além da relação com a tabela geral de personagens e das respostas para cada tipo de pergunta, contêm as respostas para as interações de acusação certa e errada, fala final caso o aluno tenha errado ou acertado as perguntas, a dica (caso haja) e se o personagem é o alvo da busca. O tutorial guarda um diálogo simples para mostrar como funciona o gameplay e os resultados, além da booleana de vitória e a variável de pontuação, guarda a contagem de perguntas válidas feitas, com quantos personagens o jogador conversou e quantas dicas ele conseguiu juntar ao final.

O jogo 3 (Assistente de mestre cuca) entra na temática de comida e foca na competência de compreender números para recolher as quantidades certas de cada item. Como ele foi planejado para o primeiro nível do curso conhecer o nome de cada alimento ainda não será o enfoque. O jogador recebera uma lista de compras a serem feitas para executar uma receita, ele deve navegar pelo mercado, pegar todos os itens necessários e então seguir para o caixa para realizar o pagamento dentro do tempo determinado. A tabela *GameThreeMissionData* guarda o tempo máximo para a execução do jogo em segundos, se os corredores devem ser divididos por tipo de comida ou não e caso não ele é informado de quantos corredores o mercado deve ter e o código separa os ingredientes aleatoriamente entre eles, o relacionamento com a tabela *Money*, que guarda todos os tipos de dinheiro disponíveis e seus assets, e a relação com a tabela de receitas.

A tabela *GameThreeRecipe* contém o nome e a descrição dos pratos com suas respectivas traduções, a imagem e a relação de ingredientes da tabela *GameThreeIngredient*. Está guarda todas as informações referentes aos ingredientes como seu nome, quantidade necessária, preço unitário, como é medido este ingrediente (ml, g ou unidade) e sua quantidade no caso de g e ml, qual o tipo do ingrediente (tempero, bebida, verdura etc.) e uma descrição que é mostrada na receita quando apresentada ao aluno. A descrição do ingrediente, que mostra a quantidade que deve ser coletada, é mostrada em português e por extenso para testar a competência desejada.

Os resultados englobam as informações de quantos segundos o aluno levou para completar o jogo, se ele executou o pagamento corretamente e quais ingredientes ele escolheu incorretamente, com o nome do ingrediente, a quantidade correta e a quantidade selecionada pelo usuário. Considera-se uma vitória se o aluno escolheu todos os ingredientes e o dinheiro corretamente dentro do tempo estipulado.

Para o jogo 4 (O prato perfeito) o objetivo passa a ser preparar a receita, selecionando os ingredientes na ordem correta e organizando as letras embaralhadas do nome, e servir o prato usando os utensílios corretos, onde ele terá que selecionar uma imagem e associar com o nome correto. O foco aqui será compreender nomes de comidas e saber quais são os talheres e louças adequados para o prato.

A estrutura para a tabela *GameFourMissionData* é a mesma do jogo anterior acrescentando a relação de utensílios. A tabela de receita também tem a mesma estrutura, mudando a relação com a tabela de ingredientes que desta vez é a *GameFourIngredient*. Esta tem o nome do ingrediente que será embaralhado, a ordem no qual ele deve ser selecionado, a relação com o asset de imagem e a descrição. Desta vez a descrição do ingrediente será realmente um passo a passo de receita e foi colocada em inglês por ser uma frase mais complexa que os alunos possam não entender. Entretanto, o nome do ingrediente continua em português (ex: “amasse o alho” -> “*mash the* alho”), assim o aluno precisa conseguir associar o nome com a imagem.

Para os utensílios tem-se a tabela *GameFourTableware*, que guarda as relações entre o asset do utensilio e da receita a qual ele vai ser associado e se ele é correto ou não. Os resultados guardam os segundos levados pelo jogador para completar o jogo, qual receita ele jogou e seus erros, ou seja, quais ingredientes ele selecionou na ordem errada, quais nomes ele montou errado, associações erradas entre nome e imagem do utensílio e utensílios inadequados. Aqui mudou-se um pouco o conceito de derrota e só é considerado assim se o aluno ultrapassar o tempo máximo de jogo.

O jogo 5 (Bem-vestido) começa a temática de roupas e busca testar se o aluno entende períodos do dia, meses do ano e datas e sabe a implicação das estações em vestimentas no Brasil. O aluno recebe um convite para um evento com a descrição de data, horário, estação e clima e terá que selecionar as roupas adequadas para ele. A *GameFiveMissionData* guarda as relações dos assets dos corpos dos personagens que o aluno pode escolher entre para jogar, das roupas e dos convites.

A tabela geral de roupas *Clothes* contém a relação com o asset (que guarda a imagem da roupa que é colocada por cima do personagem), o tipo de sobreposição da roupa (se ela vai por cima, por baixo ou cobre o corpo inteiro), a imagem da roupa quando ela está no armário e a relação com as *tags*. As *tags* guardam informações de categoria, cor, clima adequado e período do dia para a roupa, sendo essa separação feita pela variável tipo. As *tags* do tipo categoria (*category*) determinam tanto divisão das peças no armário (Pernas, Pés, Tronco e as demais são unidas em Acessórios) quanto a posição que ela fica no corpo do personagem (Cabeça, Mãos, Orelhas etc.). As de clima e período do dia são utilizadas para verificar se o usuário escolheu as roupas corretas para o evento em questão.

A *GameFiveInvite* possui as informações de mês, estação, horário e clima do evento e a relação com as *tags* consideradas corretas para as roupas para a verificação mencionada anteriormente. Os resultados armazenam a contagem de roupas que o usuário escolheu errado, se ele viu o convite novamente durante a escolha das roupas, qual personagem ele escolheu, o vestuário completo que foi selecionado e qual convite apareceu para o jogador. Considera-se derrota se o aluno escolher roupas erradas para a ocasião.

O sexto jogo (Ajude Ariel), além das competências testadas no jogo anterior, busca testar também a capacidade de discernir o vocabulário de roupas e seus possíveis adjetivos. O estudante recebe uma mensagem de Ariel que recebeu um convite para uma festa e não sabe o que vestir. Ele então deve selecionar uma roupa e em seguida descrevê-la para Ariel. Os dados desta missão são os mesmos da missão anterior sendo acrescido apenas uma variável de mensagem (*message*) no convite porque a interação entre o jogador e Ariel simula um aplicativo de mensagem no celular. Nos resultados também são acrescentadas as relações de roupas que estão no corpo e não são descritas para Ariel, roupas que foram descritas, mas não estão no corpo e roupas que foram descritas erradas.

No jogo 7 (Viagem dos sonhos) passa-se para a temática viagem, focando em competências como compreender questões de disponibilidade e agenda, saber os meses e estações do ano no Brasil, entender períodos do dia, meses do ano e datas e quantidades. O jogador interpreta um agente de viagens e recebe um e-mail de um cliente com dados para planejar a viajem ideal para ele. A *GameSevenMissionData* guarda os dados de número de dias da viagem, quantas pessoas vão, o mês da viagem e as relações com as tabelas de e-mail (*GameSevenMail*), cidades (*GameSevenCity*), voos (*GameSevenFlight*) e frases (*GameSevenPhrase*).

*GameSevenMail* tem as informações básicas de um e-mail: mensagem, data, título (com tradução) e nome e e-mail do remetente. *GameSevenCity* guarda o nome da cidade, uma imagem que é mostrada quando o usuário vai fazer a seleção e se a cidade é correta ou não dependendo do que foi pedido no e-mail do cliente. *GameSevenFlight* tem o período do voo, o tipo (ida ou volta), o horário de decolagem e aterrissagem e se o voo é uma escolha correta. Todas essas estruturas são bem simples, a tabela de frases foi a que exigiu uma construção mais complexa.

Após selecionar todas os elementos necessários, o aluno deve montar um e-mail de resposta para o cliente explicando para onde ele vai, em qual data, período e quantas pessoas vão na viagem. A oração é formada escolhendo-se entre as palavras que são mostradas para o usuário, ele deve escolher as palavras na ordem certa para que elas entreguem mensagem fazendo sentido. *GameSevenPhrase* armazena as informações para que o usuário forme essas frases. Cada frase guarda a ordem na qual ela vai aparecer, o tamanho da frase que vai ser formada, o número máximo de palavras que podem aparecer para o jogador e a relação com as palavras.

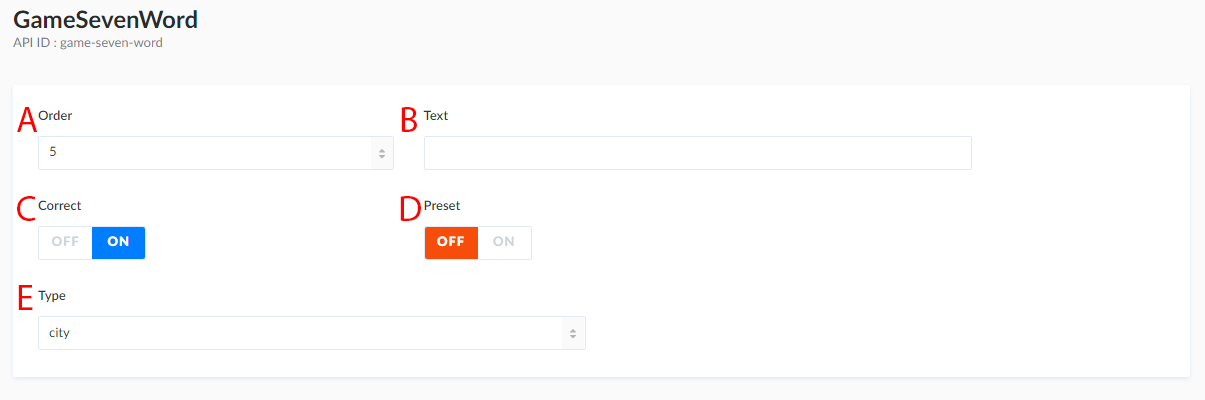
A tabela *GameSevenWord* é onde as opções de seleção são organizadas. Nela tem-se alguns condicionais. Caso a palavra esteja correta (*Figura 1* ponto C), ou seja, ela entra na frase que é enviada no e-mail, ela tem a variável ordem (*Figura 1* ponto A) que indica onde ela se posiciona na oração. Ela também pode estar predefinida ou não (*Figura 1* ponto D). Caso ela esteja, deve ser preenchido o campo de texto (*Figura 1* ponto B) com a palavra que deve aparecer. Caso contrário, a variável tipo (*Figura 1* ponto E) deve ser setada indicando com qual tipo de informação seu valor deve ser preenchido durante a execução do jogo. Exemplo disso é quando o jogador tem que construir a frase informando para qual cidade o cliente vai. Então a palavra estará com os dados como mostrados abaixo e, quando o jogo for executado, ele sabe que deve acrescentar nomes de cidade as opções de escolha do jogador. O número de palavras relacionadas à frase é sempre menor que o número máximo de opções que podem aparecer levando em conta que outras serão adicionadas durante a execução.

Figura – Estrutura da tabela GameSevenWord

Os resultados armazenam os erros que o usuário de seleção de opções e de formação de frases, mostrando o tipo do erro, qual seria a resposta correta e qual foi a resposta do jogador.

O jogo 8 (Viagem perfeita) é dividido em duas etapas. A primeira é igual ao jogo 7, mas ao invés de ter que construir o e-mail enviando as informações da viagem, na segunda o jogador recebe outras mensagens do cliente solicitando a reserva do hotel e direções para pontos próximos, como farmácias e shoppings, de acordo com um mapa mostrado com as construções em questão. Assim, este jogo também pretende testar se o aluno é capaz de fazer reservas em hotel, discernir locais pela descrição e interpretar direções. A tabela *GameEightMissionData* também guarda os dados de número de dias da viagem, quantas pessoas vão, o mês da viagem e as relações com as tabelas de e-mail (*GameEightEmail*), cidades (*GameEightCity*) e voos (*GameEightFlight*). Além destes há os dados de locais (*GameEightLocation*) que serão mostrados no mapa que aparece para o jogador.

A estrutura da tabela *GameEightEmail* contém os dados básicos de um e-mail como a *GameSevenMail*, unificando os dados que são comuns entre todos os e-mails que são recebidos, mas ao invés de ter uma única mensagem, ela possui uma relação com a tabela *GameEightEmailBody* que guarda a mensagem que é mostrada, se o e-mail é mostrado quando o jogador seleciona as opções corretas ou não, pra qual tipo de resposta o e-mail é usado (quando o usuário selecionou um hotel, os voo etc.) e em qual ordem ele deve ser mostrado (caso haja mais de uma mensagem de um mesmo tipo). Caso o e-mail do cliente peça instruções de chegada em algum ponto do mapa, o aluno deverá montar uma mensagem com as informações necessária.

A estrutura para a construção das respostas se encontra na variável *responseEmail* que é uma relação com a tabela *GameEightResponseEmail*. Ela possui uma mensagem, que é o texto inicial que será mostrado no e-mail, o tamanho máximo que a frase formada pelo aluno pode ter e a relação com a tabela *GameEightResponseEmailPartial*, que guarda uma instrução para a construção da frase (ex: “vire à direita”, “siga em frente”), se ela é correta ou não e, caso ela seja correta, a ordem na qual ela se encaixa na formação da frase.

A tabela *GameEightCity*, além das informações da tabela *GameSevenCity*, guarda a imagem do mapa cidade que possui a mesma organização base de ruas para todas as cidades mudando apenas alguns detalhes de pontos turísticos e prédios assinatura de cada uma. *GameEightLocation* representa um ponto que é mostrado no mapa, armazenando seu nome, uma descrição, o tipo de estrutura (escola, hospital, hotel etc.), um ícone que é mostrado no mapa e representa o tipo, a posição x e y em relação ao canto superior esquerdo para a colocação do ponto no mapa e uma variável booleana que indica se o local é correto ou não considerando a situação quando o jogador deve fazer a reserva no hotel pedido pelo cliente. Os resultados são guardados seguindo a mesmo arranjo da tabela de resultados do jogo anterior.

No jogo 9 (Corretor sinistro) o aluno é um corretor de imóveis e terá uma fila de clientes para atender dentro de um tempo determinado. Cada cliente chega pedindo imóveis com determinadas características e o jogador deve escolher uma casa que satisfaça as condições exigidas. A tabela *GameNineMissionData* guarda o tempo máximo de jogo em segundos, a imagem do mapa da cidade onde serão posicionados os elementos e as relações de personagens, casas (*GameNineHome*), locais (*GameNineLocation*) e pedidos (*GameNineRequest*).

Dos personagens são utilizadas apenas as imagens, os diálogos dos pedidos foram pensados para serem independentes de gênero e, assim, possibilitar que os personagens sejam escolhidos aleatoriamente cada vez que o aluno entre no jogo para que ele não decore a ordem dos pedidos ou dos personagens já que o tempo influencia no resultado do jogo. A tabela *GameNineRequest* tem todas as informações relacionadas ao pedido que é feito ao jogador. Nela estão o diálogo inicial que é apresentado, quantos quartos e/ou banheiros deve haver na casa, se a casa deve ser perto ou longe de algum referencial, dificuldade do pedido (número de variáveis [quartos, banheiros, perto de, longe de] que descrevem a casa – de 1 a 4) e diálogos de erro, ou seja, o personagem terá ao menos uma frase para cada variável que a casa selecionada não cumprir.

As tabelas *GameNineHome* e *GameNineLocation* descrevem objetos que são localizados no mapa. *GameNineHome* é responsável pelas informações das moradias com o nome do condomínio, número de quartos e banheiros, a quais locais ele é próximo, ícone e posição X e Y para ajuste no mapa. Já a *GameNineLocation* contém informações sobre os demais locais que devem aparecer no mapa como comércio, hospitais e escolas. Ela contém uma descrição do local, o tipo, um ícone e a posição X e Y.

**Jogos**

Aqui será feita uma descrição mais aprofundada da jogabilidade e como os dados são recebidos do banco de dados e adaptados para cada jogo.

**Jogo 1 – Crise de meia idade**



Figura 2 - Tela principal de sala do jogo “Crise de meia idade”

No jogo “Crise de meia idade” o jogador deve verificar sua lista de contatos, disponível no ícone de celular no canto inferior direito da tela (*Figura 2* ponto C), e identificar quais contatos têm informações incompletas (*Figura 3*). Para descobrir os dados faltantes, ele deve abrir uma conversa com os personagens presentes na sala na qual ele se encontra clicando sobre o personagem desejado (*Figura 2* ponto B), iniciar um diálogo escolhendo um cumprimento dentro das opções disponíveis (*Figura 4*) e então escolher as perguntas que darão a ele a resposta para completar o contato (*Figura 5*). Para isso, o jogador tem um limite de perguntas que ele pode fazer (*Figura 1* ponto A) em cada uma das salas que, quando alcançado, impede-o de continuar fazendo novas perguntas (*Figura 6*).



Figura 3 - Tela de lista de contatos

O jogo foi projetado de forma que a quantidade de informações a serem preenchidas vai aumentando conforme o aluno vai avançando pelas salas, começando com apenas um dado incompleto de um contato (sala 1) e na última sala (sala 5) faltam oito informações (duas de cada personagem). O número de perguntas que o usuário pode fazer por sala também varia para equilibrar.

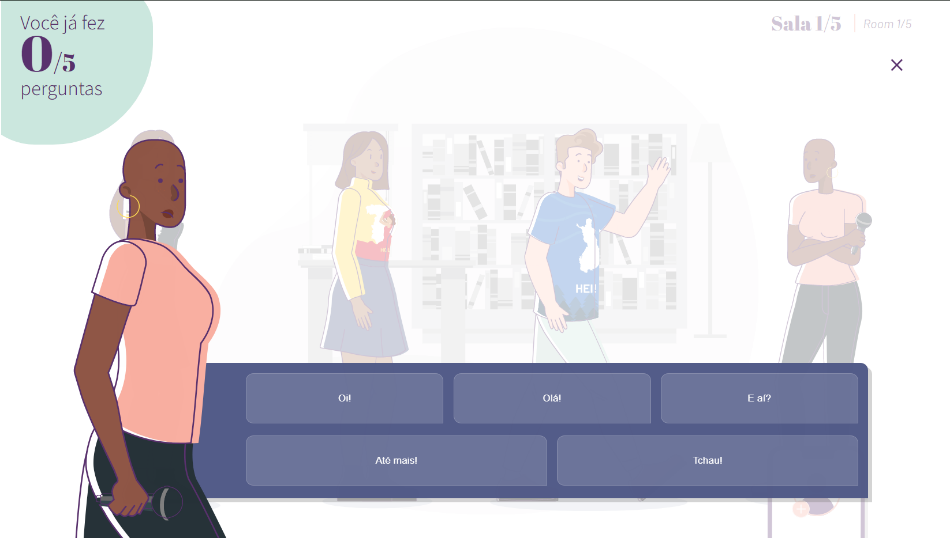


Figura 4 - Tela de início de diálogo

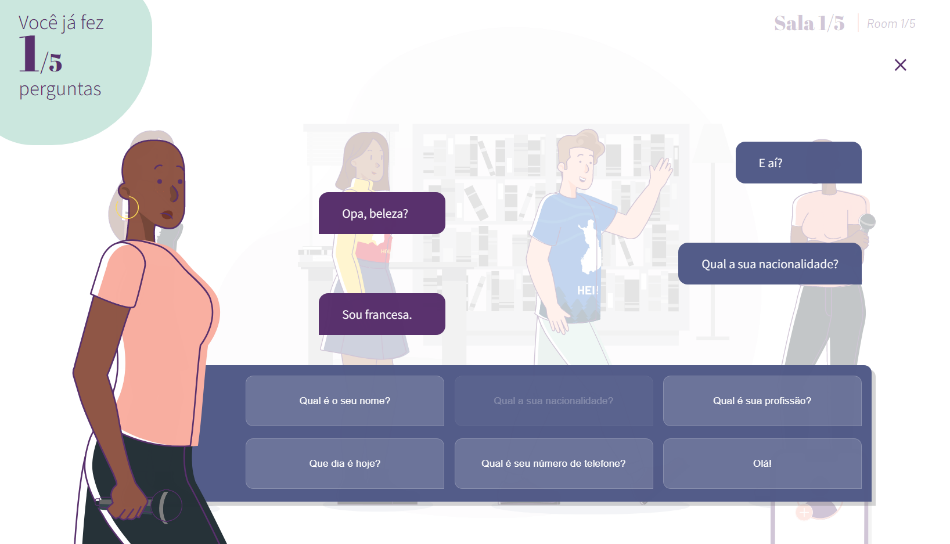


Figura 5 - Tela de conversação

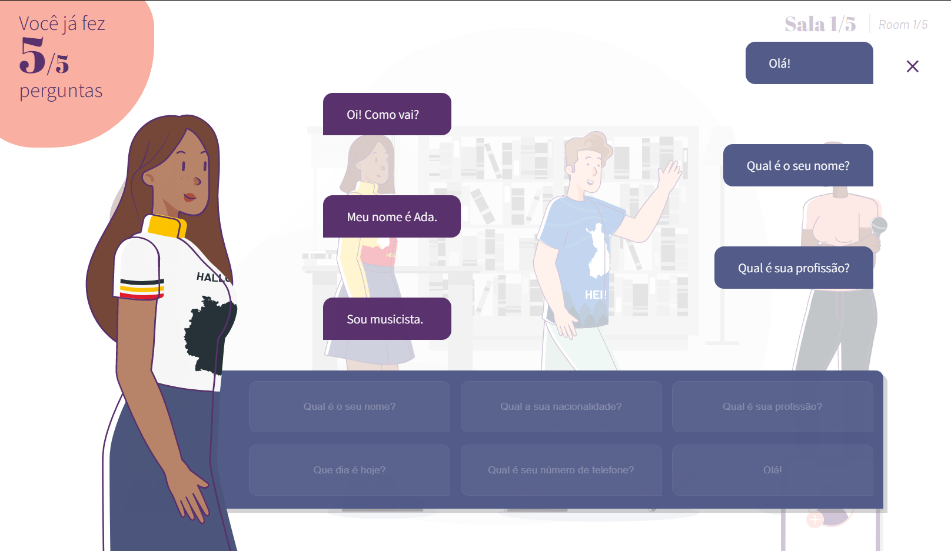


Figura 6 - Limite de perguntas alcançado

Cada contato possui campos de nome, profissão e país e cada uma dessas informações é contemplada com uma pergunta disponível na conversação (*Figura 5*). Tanto na parte de cumprimento quanto na parte de conversação existem opções de diálogo fora de contexto ou erradas. Isso, junto com o limite de perguntas, forma um sistema de punição para cada vez que o aluno escolhe uma opção aleatoriamente ou apenas não sabe qual escolher para obter a resposta desejada.

A opção de avançar para a próxima sala só aparece quando o jogador preenche todas os dados que faltam em cada contato e o jogo emite um aviso caso alguma delas esteja incorreta. O aluno pode então escolher se ele vai seguir para a próxima sala ou se continua a tentar encontrar a resposta certa. Ao final o aluno receberá diferentes *feedbacks* dependendo de quantas informações corretas ele conseguiu coletar, sendo considerada uma vitória uma porcentagem de acertos maior que 80%.



Figura 7 – Página de resultado do jogo

Na tela de apresentação de cada um dos jogos, o jogador só pode clicar para dar início ao jogo ao final do carregamento dos dados. Esse carregamento é o processo em que o jogo recebe os dados do banco e os organiza para que o código consiga lê-los. Ao receber as informações do banco, a primeira coisa a ser feita é organizar as salas a partir da variável *order*, para que elas fiquem na ordem correta que o jogador deve seguir, e aleatorizar a ordem em que os personagens se encontram dentro do mesmo ambiente, para que a organização dos personagens dentro da sala e na lista de contatos não seja a mesma.

Em seguida são coletadas todas as alternativas possíveis de cada dado dos contatos para serem colocados como opções na lista de seleção que aparece nos campos a serem preenchidos. Por fim é criado o gabarito com todas as informações corretas de cada personagem para verificação do resultado além de uma versão do gabarito retirando os campos que não devem ser mostrados (seguindo as variáveis *showCountry*, *showJob* e *showName*). Essa versão será utilizada na lista de contatos para o preenchimento das informações pelo usuário e na contagem total de campos vazios, para o cálculo da porcentagem de acerto.

**Jogo 2 – Um novo começo**

O contexto que o aluno se encontra no jogo “Um novo começo” é o de estar começando em um novo emprego e ele deve encontrar a pessoa responsável por guiá-lo pelo local. O jogador terá um diálogo introdutório com um NPC (*Non-player character* – personagem não jogável) para perguntar quem é essa pessoa que deve ser encontrada. O NPC responde que não sabe quem é a pessoa, dá uma dica de alguém que possa saber e o jogador é direcionado para a primeira sala (*Figura 8*). Essa interação inicial é um recorte da mecânica geral do jogo que é interagir com os personagens presentes nas salas e conseguir dicas de características físicas ou informações pessoais restringir as possibilidades de escolha.



Figura 8 – Tela principal de sala do jogo “Um novo começo”

O usuário pode navegar pelas salas utilizando os botões no canto superior direito da tela (*Figura 8* ponto B) e escolher com qual personagem deseja conversar. Não existe uma ordem pré-definida para os diálogos, entretanto, as dicas dadas indicam uma possível lógica a ser seguida. Ao clicar sobre o NPC (*Figura 8* ponto C), um novo diálogo será aberto com duas opções de perguntas para serem feitas (*Figura 10*), uma certa (que se encaixa no contexto da conversa e que ajuda a identificar se as dicas se referem ao personagem com quem o jogador está conversando) e outra errada (fora de contexto e que não traz nenhuma informação útil).

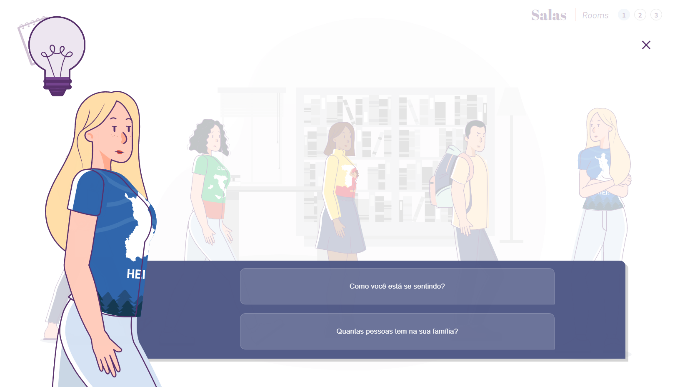


Figura 10 – Tela inicial de conversa

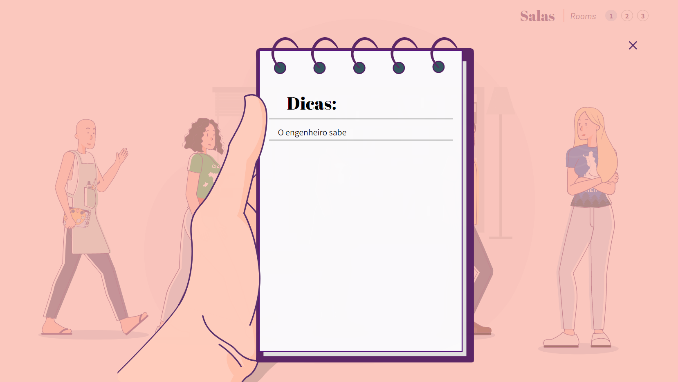


Figura 9 – Tela de dicas

O jogador deve escolher duas perguntas, ou seja, ao abrir a tela aparecem duas opções, ele escolhe uma, o personagem responde, em seguida aparecem mais duas opções de pergunta e o processo se repete. Caso ele escolha duas opções corretas (*Figura 11*), há uma chance de que o NPC possua uma dica e, se tiver, ela será passada para o aluno e ficará disponível na lista de dicas, que pode ser visualizada clicando no ícone de bloco de notas no canto superior direito quando o jogador está fora da tela de conversa (*Figura 8* ponto A e *Figura 9*). Caso ele escolha pelo menos uma alternativa errada, o NPC perguntará se o jogador gostaria de recomeçar a conversa (*Figura 12*) para que o jogador possa tentar novamente. Ao total existem sete dicas espalhadas entre quinze personagens e uma dica dada na abertura do jogo.



Figura 12 – Conversa finalizada selecionando alternativas incorretas



Figura 11 – Conversa finalizada selecionando alternativas corretas

Quando o aluno tiver um palpite de quem é a pessoa que procura, ele deve abrir uma conversa com o personagem desejado e clicar no ícone de lâmpada no canto superior esquerdo (*Figura 12* ponto A). Será aberto então um *pop-up de* confirmação com todas as dicas obtidas até o momento pelo jogador (*Figura 13*) para que ele possa rever os dados antes de confirmar sua escolha. A tela de resultado mostra se ele acertou ou não a pessoa e as estatísticas de jogo (com quantas pessoas ele falou, quantas dicas ele obteve e quantas das perguntas feitas por ele foram corretas).



Figura 13 – Pop-up de confirmação de acusação

Quando os dados chegam ao jogo, é criada uma variável *locations* que guarda a imagem de fundo, o nome da sala e todos os personagens que ficam nela. Os personagens são distribuídos aleatoriamente entre as salas para que cada vez que o aluno entre no jogo os personagens estejam em posições diferentes. Além disso são aleatorizadas também quais perguntas vão aparecer e qual a ordem delas para cada um dos NPCs. Ao todo existem seis alternativas de perguntas, três certas e três erradas. Por fim é feita a contagem total de dicas para estatísticas da tela de *feedback*.

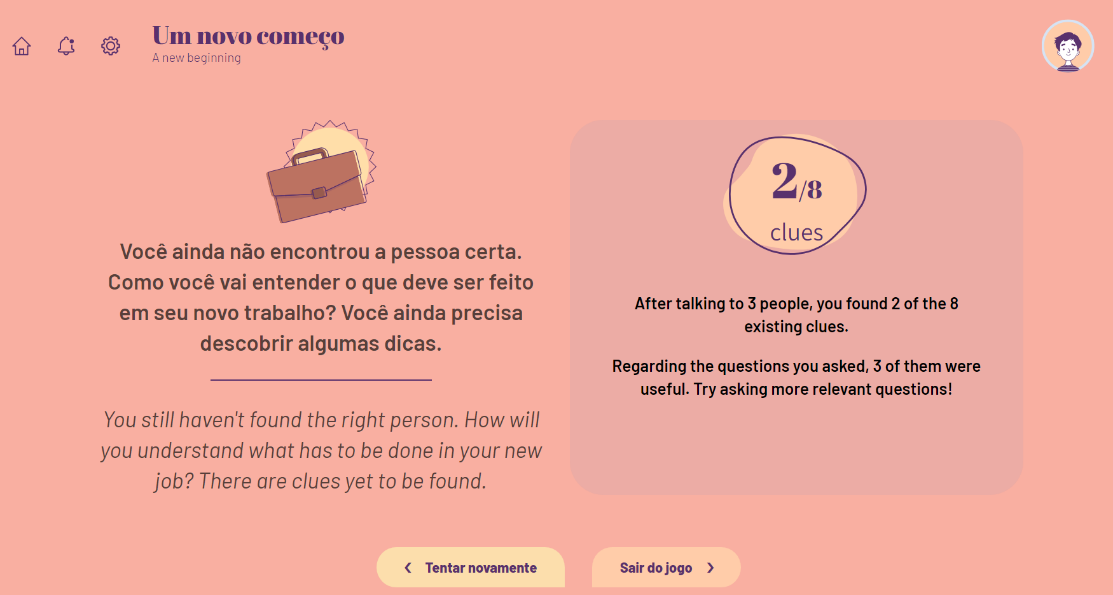


Figura 14 – Tela de resultado de derrota

**Jogo 3 – Assistente de Mestre Cuca**

O terceiro jogo, “Assistente de Mestre Cuca”, é voltado para a culinária brasileira. Ele começa com uma introdução textual em formato de diálogo com uma chef explicando o objetivo do jogo, quais são os ingredientes que devem ser coletados para preparar um prato e dando uma pequena explicação do contexto do prato na cultura do país (*Figura 15*). Esse jogo é o primeiro em que o aluno tem um tempo máximo para chegar ao final. Então, antes do início da *gameplay,* é mostrada ao jogador a lista de ingredientes da receita (*Figura 16*) para que ele possa analisar cada um dos elementos com mais calma e é feito um tutorial completo de cada componente da interface com suas funcionalidades.



Figura 16 – Apresentação da receita

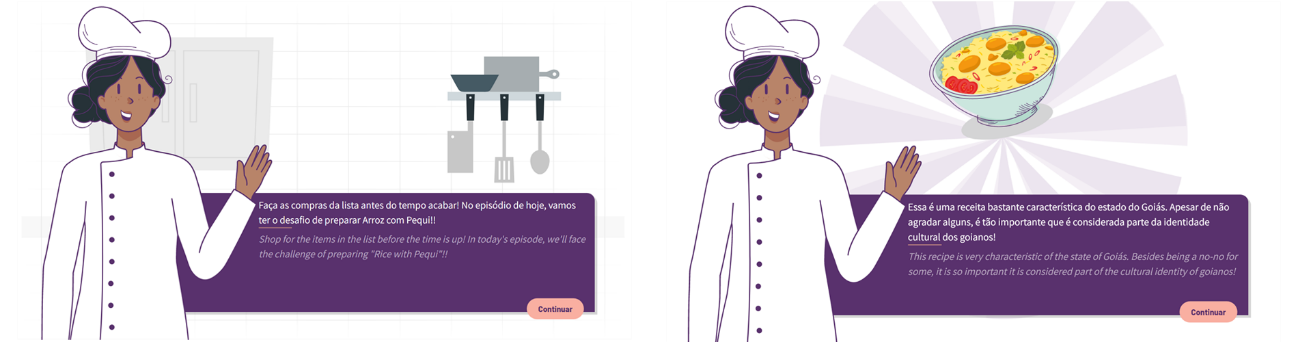


Figura 15 – Telas de diálogo de introdução

O ambiente do jogo é um mercado (*Figura 17*) e o objetivo inicial do aluno é coletar todos os ingredientes da lista na quantidade correta. O cenário mercado é dividido pelo que foi chamado de corredor, são três no total, cada um contém três prateleiras com um tipo de ingrediente (*Figura 17* ponto E) e, para alternar entre os corredores, o jogador deve clicar nas setas posicionadas nas laterais (*Figura 17* pontos Ds). Ao clicar sobre um ingrediente da prateleira (*Figura 17* ponto F), ele é adicionado ao carrinho (*Figura 17* ponto G) e sua imagem fica exibida nesse espaço com um ícone numérico que indica quantos desse mesmo elemento já foram colocados ali. Para retirar um ingrediente do carrinho, o usuário deve aplicar o clique sobre a imagem que estiver dentro do espaço correspondente ao carrinho de compras (*Figura 17* ponto G).

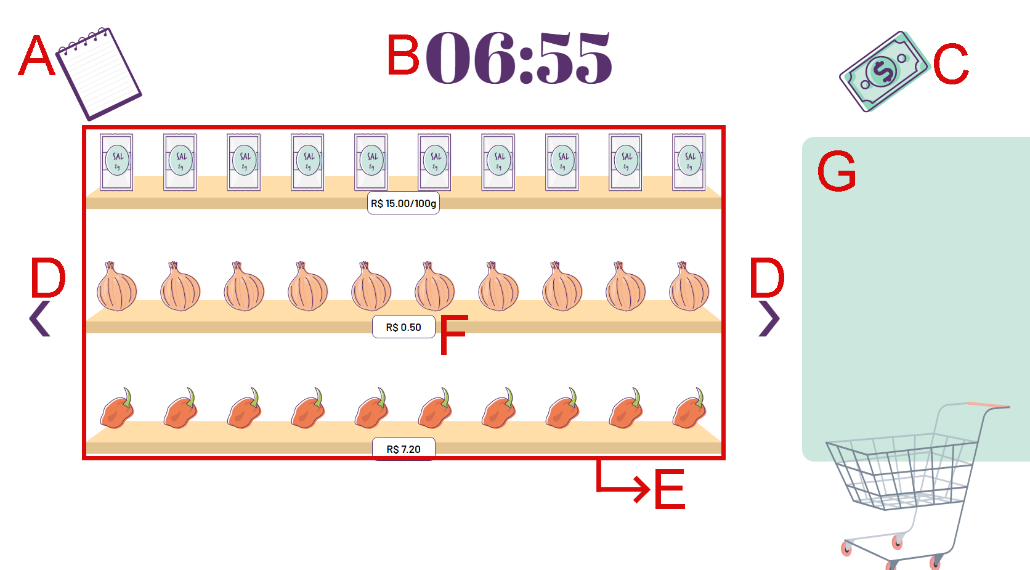


Figura 17 – Tela de mercado

O ícone de bloco de notas, localizado no canto superior esquerdo (*Figura 17* ponto A), é responsável por exibir a lista de ingredientes quando o jogador posiciona o mouse sobre ele (*Figura 18*). Ao lado de cada um dos itens da lista há uma caixa de seleção que é automaticamente marcada sempre que a quantidade correta daquele determinado ingrediente é coletada (*Figura 19* ponto A). Quando o jogador quiser seguir para a próxima etapa, ele deve clicar no ícone de cédula (*Figura 17* ponto C), entretanto, caso não tenha pegado todos os ingredientes corretamente, ele recebe uma mensagem do NPC e é mandado de volta para a etapa do mercado. Isso ocorrerá até que todos os itens sejam cumpridos ou que o tempo acabe. O tempo restante de jogo é exibido centralizado no topo da tela (*Figura 17* ponto B).

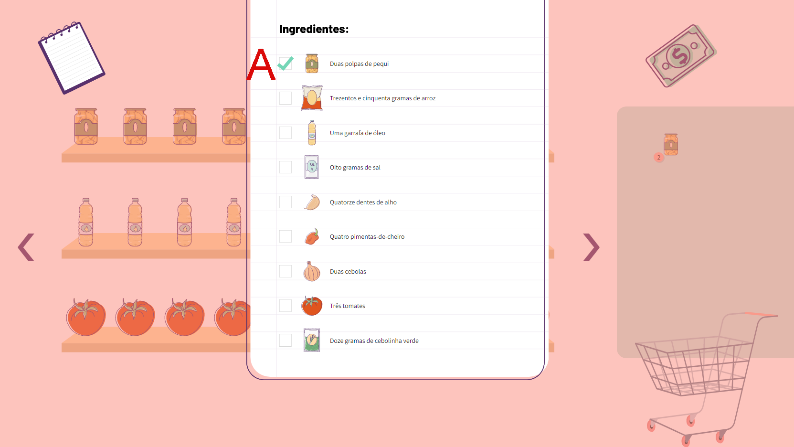


Figura 19 – Receita após a adição da quantidade correta de um ingrediente

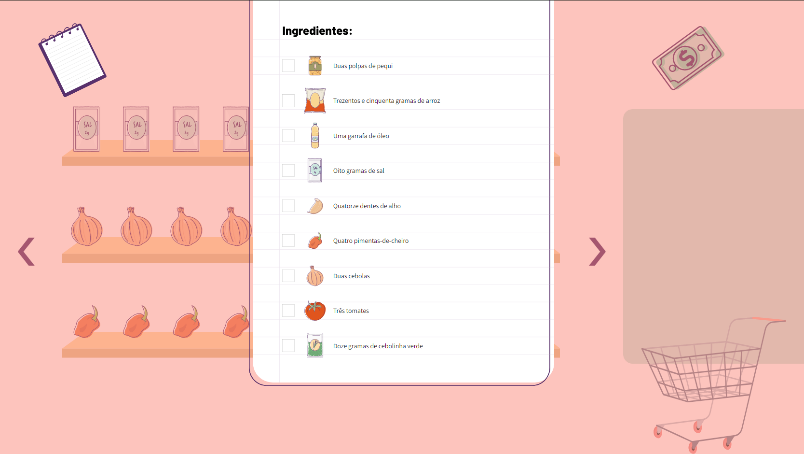


Figura 18 – Tela de visualização da receita

Na próxima fase é exibido o valor total da compra por extenso e o aluno deve utilizar o tempo que restou após as compras para efetuar o pagamento. Para a transformação do valor numérico da conta para texto, foi feita uma adequação da função *humanize* (encontrada no link <https://gist.github.com/ForbesLindesay/5467742>) em que houve a tradução para o português e uma adaptação para que a função suportasse números de ponto flutuante e para que ao final de cada conversão fossem adicionadas as unidades de “real” ou “centavos”.

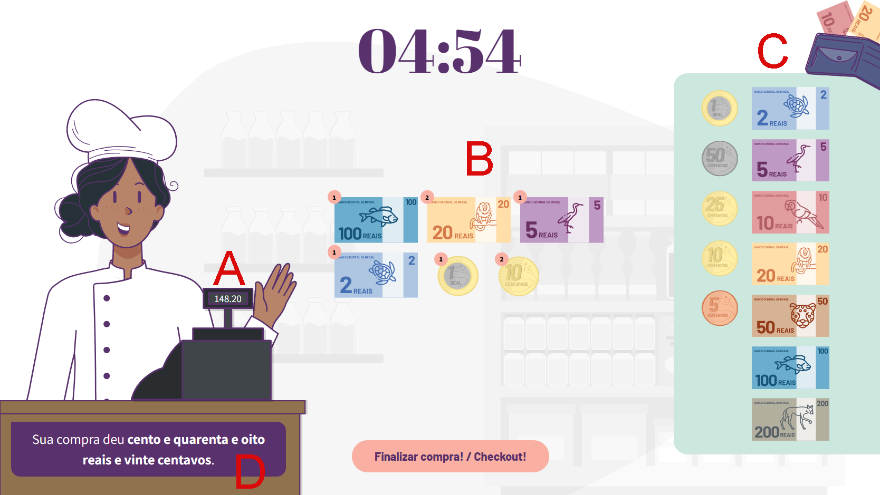


Figura 20 – Tela de pagamento das compras

Na tela de caixa são exibidos todas as atuais cédulas e moedas brasileiras e o aluno deve fazer uso delas para chegar ao valor estipulado. Aqui tem-se uma mecânica similar a do mercado, onde um clique sobre o dinheiro da carteira (*Figura 20* ponto C) o adiciona ao pagamento e o clique sobre o dinheiro na área de pagamento (*Figura 20* ponto B) o remove. O valor da somatória do dinheiro na área de pagamento fica exibido no visor do caixa (*Figura 20* ponto A) e o valor total da compra também é sempre mostrado por extenso em uma caixa de texto a frente da mesa do caixa (*Figura 20 ponto D*). Para finalizar a transação basta clicar no botão “Finalizar compra! / Checkout!”.



Figura 21 – Tela de resultado

Por fim o NPC exibe último diálogo que varia de acordo com o pagamento que foi feito (se foi abaixo, acima ou igual ao valor da compra) e então o aluno é encaminhado para a tela de resultado do jogo. Aqui só é considerada uma derrota quando o tempo se esgota, caso contrário é exibida a tela de vitória do jogo com quanto tempo o jogador levou para concluir a percurso (*Figura 21*).

A organização das informações vindas do banco de dados começa com a criação de duas variáveis para guardar o tempo total de jogo e o tempo restante, ambos em segundos. Em seguida é selecionada aleatoriamente dentre três possibilidades a receita que será apresentada no jogo e feita a descompactação dos objetos de ingredientes para facilitar a leitura pelo código. Por último é feita a separação dos ingredientes entre os corredores de maneira aleatória.