SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE VIÇOSA · UFV
CAMPUS FLORESTAL

# Design CSU07 e CSU08

## Bytecraft - 5º ano

Florestal - MG
2025

**Sumário**

# Backend CSU07 e CSU08

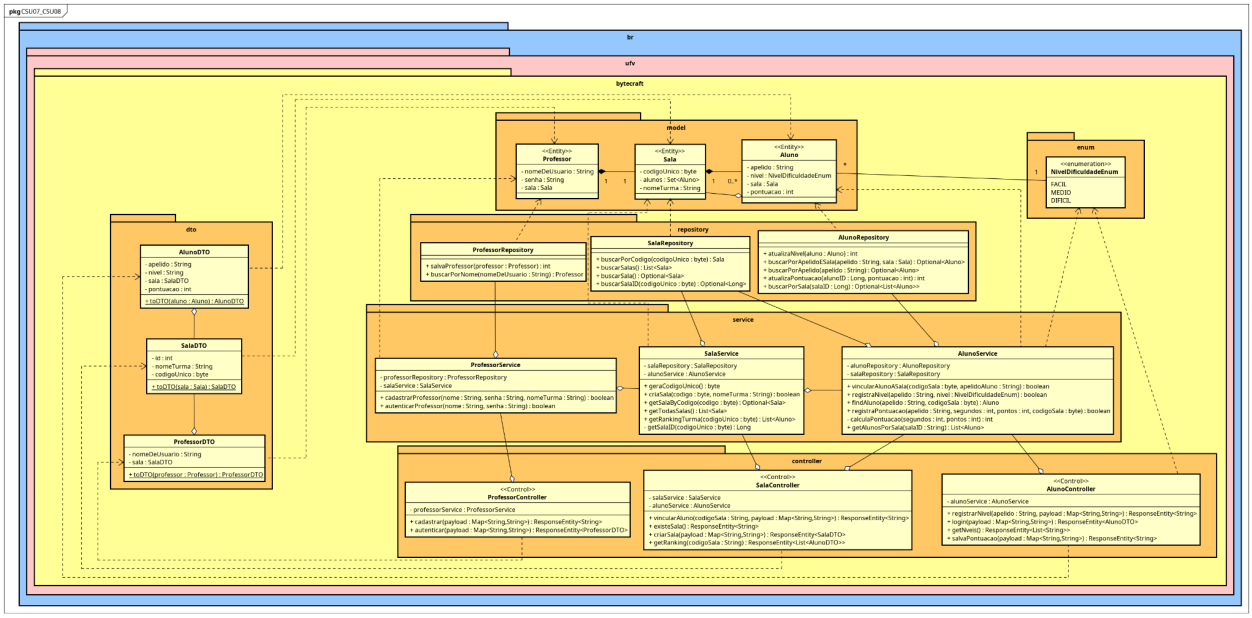


**Figura 1**: Diagrama de classes com CSU07 e CSU08.

A classe de entidade **Aluno** obteve um novo atributo privado **pontuacao**, do tipo **int**, com seus respectivos getters e setters, que deve ser mapeado no banco de dados para armazenar a pontuação total do aluno.

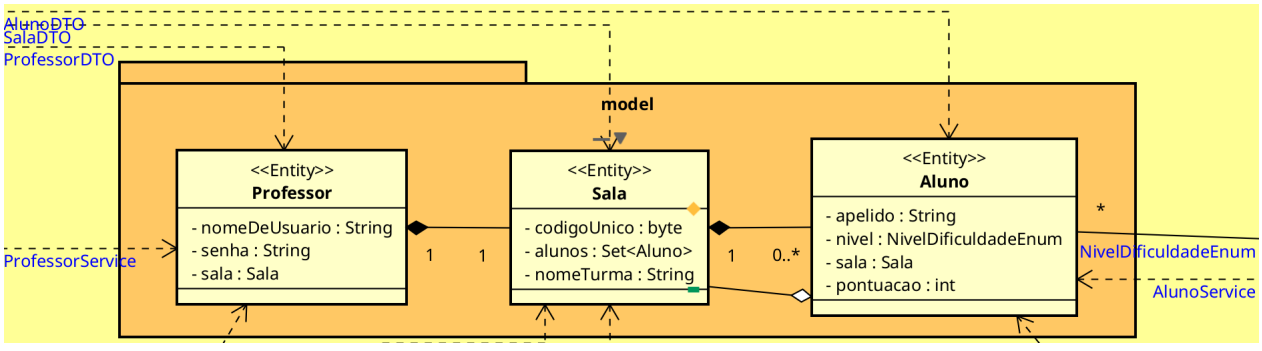


**Figura 2**: Classes de entidade.

A classe **AlunoRepository** deve implementar dois novos métodos, o primeiro é o método **atualizaPontuacao**, que será responsável por executar a atualização da coluna de pontuação para um aluno específico. O segundo método é o **buscarPorSala**, que deve receber um ID de Sala

como parâmetro e retornar um **Optional<List<Aluno>>**, sendo sua função consultar todos os alunos que estão vinculados a uma determinada sala.

A classe **SalaRepository** deve implementar um novo método de consulta chamado **buscarSalaID**, que recebe um **codigoUnico** como parâmetro e é responsável por retornar o ID correspondente à entidade Sala em questão.

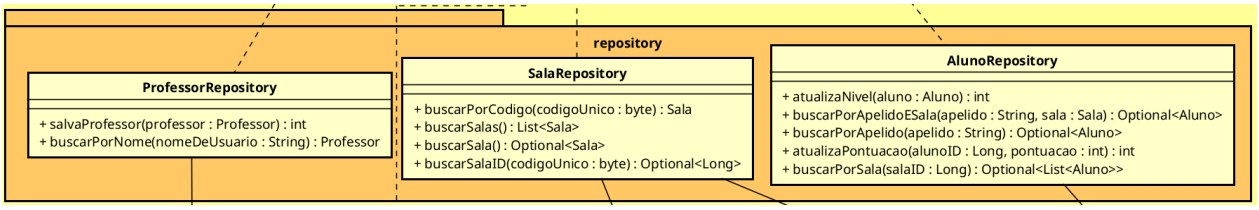


**Figura 3**: Classes de comunicação com o banco de dados.

A classe **AlunoService** deve ser estendida para conter a nova lógica de negócio relacionada à pontuação e listagem de alunos. Ela deve implementar o método **registraPontuacao**, responsável pela persistência da pontuação de um aluno, além de chamar o método **calculaPontuacao** para calcular a pontuação total. Adicionalmente, deve implementar o método **calculaPontuacao**, que conterá o algoritmo de acordo com a RN22 para determinar a pontuação de um aluno. Por fim, a classe deve ter o método **getAlunosPorSala**, que invocará o método **buscarPorSala** do repositório para obter e retornar a lista de alunos de uma turma.

A classe **SalaService** obteve um novo atributo privado, o **alunoService**, do tipo **AlunoService**, que deve ser injetado via construtor para permitir que a lógica de sala possa acessar e utilizar funcionalidades relacionadas ao aluno, como a busca de participantes de uma turma. A classe também deve implementar o método privado **getSalaID**, que servirá como uma camada de abstração para invocar o método **buscarSalaID** do **SalaRepository**. Para a funcionalidade de ranking, a classe deve implementar o método **getRankingTurma**, que será responsável por utilizar o **alunoService** para buscar os alunos de uma sala, ordená-los com base no atributo **pontuacao** e retornar a lista classificada, usando o método **getSalaID** no processo.

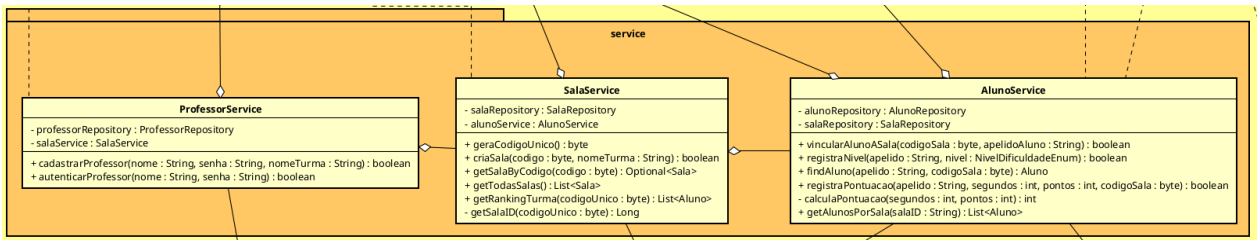


**Figura 4**: Classes de serviço.

A classe **AlunoController** deve implementar um novo método, o **salvaPontuacao**, que será exposto como um endpoint na API no caminho “api/alunos/setPontuacao”. Este método será responsável por receber as requisições do frontend para registrar uma nova pontuação para um aluno, devendo invocar o método **registraPontuacao** da classe **AlunoService** para executar a operação.

A classe **SalaController** deve implementar o novo método **getRanking**, que funcionará como o endpoint da API no caminho “api/salas/getRanking/{codigoUnicoDaSala}” para a funcionalidade de ranking. Este método receberá uma requisição para obter o ranking de uma turma e será responsável por invocar o método **getRankingTurma** da classe **SalaService**, retornando em seguida a lista de **AlunoDTO** ordenada por pontuação para o frontend.

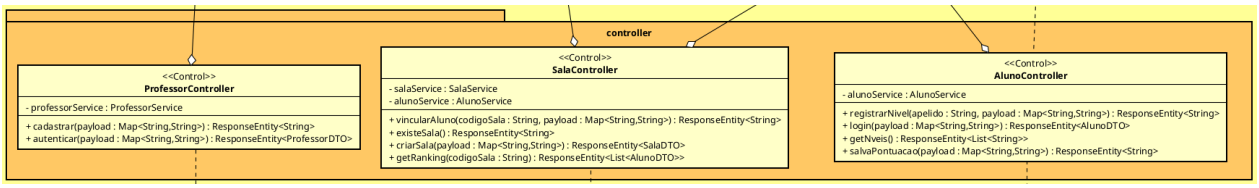


**Figura 5**: Classes de controle.

A classe **AlunoDTO** deve espelhar a mudança da entidade Aluno, recebendo também um novo atributo privado **pontuacao**, do tipo **int**, para que seja possível transportar a informação de pontuação do aluno através da API de forma segura.

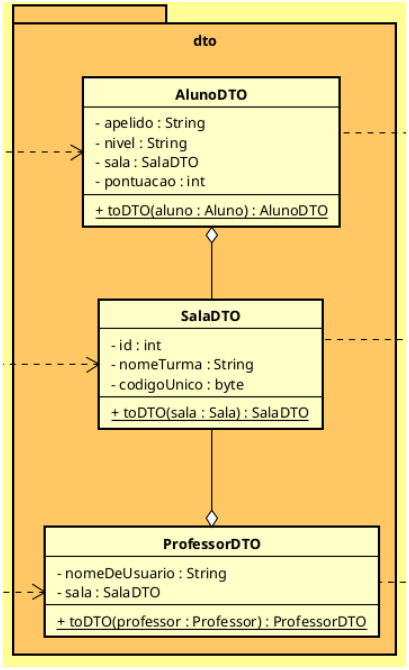


**Figura 6**: Classes de DTO.

# CSU07

**Navegar para a tela de fases (CSU07)**

**Sumário**: Permite que o aluno acesse a tela de fases após se vincular à sala, onde poderá escolher entre os modos de jogo disponíveis.

**Ator primário:** Aluno

**Precondições:**

1. O aluno já deve estar autenticado e corretamente vinculado a uma sala existente.
2. O aluno já deve ter passado pela tela de níveis e selecionado a opção desejada.
3. O progresso do aluno (se já jogou o modo História) deve estar registrado no sistema.

**Fluxo Principal**

1. O aluno, após se autenticar, é redirecionado para a tela de fases, que exibe os botões: Modo História e Modo Quiz.
2. O sistema carrega os dados do progresso.
3. O aluno clica no botão Modo História.

4. O sistema inicia o Modo História.
5. Após concluir o modo História, o aluno retorna à tela de fases.
6. O aluno clica no botão Modo Quiz.
7. O sistema inicia o Modo Quiz.
8. Após concluir o Modo Quiz, o aluno tem a opção de retornar à tela de fases ou verificar o Ranking da turma.

**Fluxo Alternativo (2)**

1. O sistema não consegue recuperar os dados de progresso do aluno:
    a. O sistema exibe a mensagem:
    "Não foi possível carregar seu progresso. Tente novamente mais tarde."
    b. O aluno permanece na tela de fases.

**Fluxo Alternativo (3)**

1. O aluno tenta clicar em Modo Quiz antes do Modo História mas não é seu primeiro acesso e ele já concluiu o Modo História.
    a. O sistema permite o acesso ao Modo Quiz.
    b. O fluxo alternativo retorna ao 7 do fluxo principal.

**Fluxo Alternativo (5 e 6)**

1. O aluno tenta clicar em Modo Quiz, mas é seu primeiro acesso ou ainda não concluiu o Modo História.
    a. O sistema bloqueia o acesso ao Modo Quiz.
    b. O sistema exibe a mensagem:
    "Você precisa concluir o Modo História antes de acessar o Modo Quiz."
    c. O aluno permanece na tela de fases.

**Pós-condições**

(Sucesso): O aluno conseguiu acessar a tela de fases e iniciar pelo menos um dos modos de jogo permitidos, de acordo com os dados de seu progresso carregados corretamente pelo sistema.
(Falha): O aluno não pôde acessar nenhum dos modos de jogo devido à falha no carregamento dos dados de progresso, permanecendo assim na tela de fases sem conseguir interagir com os modos.

**Regras de Negócio**

RN08: O aluno somente pode acessar o Modo Quiz após concluir o Modo História. Se não tiver

concluído, o botão deve ser bloqueado e uma mensagem deve ser exibida.

RN09: Sempre que o aluno for redirecionado para a tela de fases, o sistema deve verificar se é seu primeiro acesso; se já concluiu o Modo História; Para então determinar o estado (ativo/inativo) do botão "Modo Quiz".

RN10: Se o aluno tentar acessar o Modo Quiz sem ter concluído o Modo História, o sistema deve impedir a navegação e exibir uma mensagem de erro.

RN11: Se o sistema não conseguir carregar o progresso do aluno, ele deve exibir uma mensagem de erro.

## Frontend

Deve ser criada 1 página: a página de Fases. Deve ser seguida como base de implementação a figura abaixo.


**Figura 7**: Tela de Fases.

O fluxo das telas deve ser o seguinte, como pode ser visualizado na Figura 7:

- **Tela de Fases**
    - Esta tela representa o início do fluxo após o aluno selecionar o nível de dificuldade.
    - Com base no atual momento da aplicação:
        - Se o Modo História já foi concluído, o botão “Modo Quiz” deve estar ativo.
        - Se o Modo História não foi concluído, o botão “Modo Quiz” deve estar desativado/bloqueado.

- Ao clicar no botão “Modo História”, o sistema deve redirecionar o aluno para a tela “Modo História”.
- Ao clicar no botão “Modo Quiz” (se estiver ativo), o sistema deve redirecionar o aluno para a tela “Modo Quiz”.
- Em caso de falha ao carregar o progresso, deve-se exibir a mensagem de erro especificada em CSU07 ("Não foi possível carregar seu progresso. Tente novamente mais tarde.").
- Se o aluno clicar no botão “Modo Quiz” enquanto ele estiver desativado, deve-se exibir a mensagem de erro especificada em CSU07 ("Você precisa concluir o Modo História antes de acessar o Modo Quiz.").

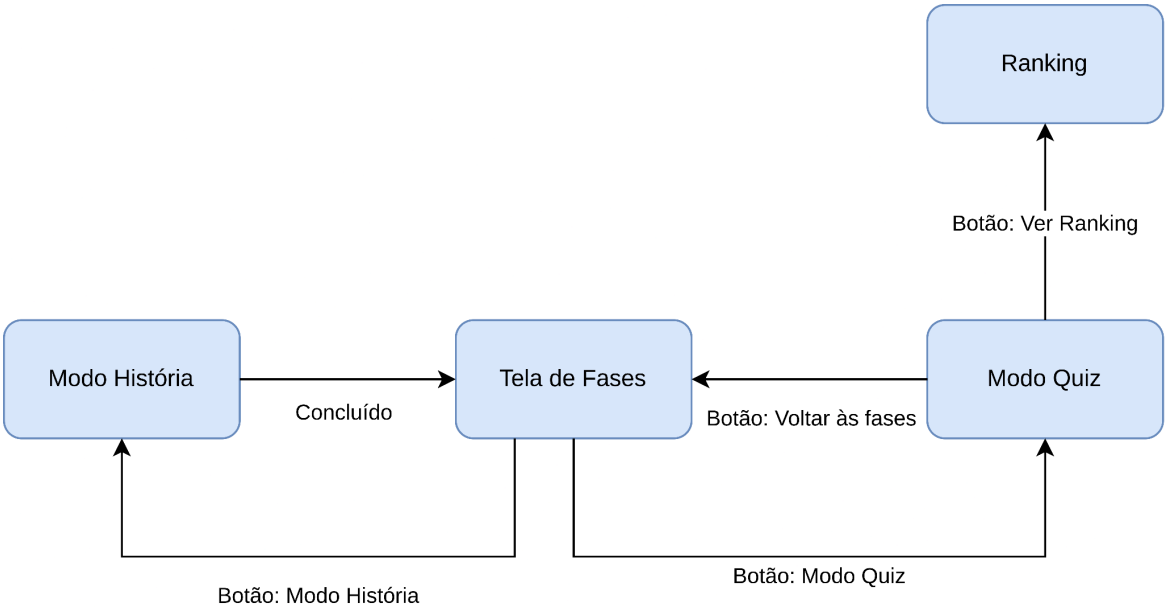


**Figura 8**: Diagrama representando o fluxo de telas de CSU07.

Espera-se que o sistema seja capaz de carregar corretamente o progresso do aluno, ajustar dinamicamente a interface com base nesse progresso, redirecionar o aluno para os modos de jogo correspondentes e tratar erros de forma clara, garantindo que as regras de acesso aos modos de jogo sejam respeitadas.

# CSU08

**Posicionar Peça na Montagem (CSU08)**

**Sumário**: O aluno usa o sistema para posicionar uma peça na montagem de um computador.

**Ator primário:** Aluno.

**Precondições:** O aluno pertence a uma sala e selecionou o Modo História.

**Fluxo Principal**

1. O aluno acessa o Modo História na tela de fases.
2. O sistema apresenta a primeira história do modo explicando sobre o funcionamento da peça e onde ela se encaixa
3. O sistema exibe uma lista de peças disponíveis para a montagem.
4. O aluno seleciona uma das peças disponíveis.
5. O sistema carrega a área de encaixe.
6. O aluno seleciona uma área do computador onde acredita que a peça deve ser encaixada.
7. O sistema verifica se a posição escolhida é a correta e computa a pontuação do aluno de acordo com a RN10.
8. O sistema posiciona a peça no local e exibe visualmente a montagem com a peça encaixada.
9. Após a última história, o sistema exibe uma mensagem sinalizando que aquele modo foi concluído e calcula a pontuação final do aluno de acordo com a RN22.
10. O aluno pode retornar novamente para a tela de fases.

**Fluxo Alternativo (7)**

1. Se a área selecionada é incorreta
    a. O sistema exibe mensagem de erro informando que a peça não pode ser encaixada na área selecionada.
    b. O fluxo retorna para o passo 5 do fluxo principal.

**Pós-condições**

(Sucesso) A peça é posicionada corretamente no local apropriado da montagem, a montagem do computador avança com a inclusão da nova peça, e o sistema registra o progresso e a pontuação do aluno.

(Falha) Nenhuma alteração é feita na montagem, e o aluno é notificado do erro e instruído a

corrigir sua ação.

**Regras de Negócio**

RN12: Para tentar o encaixe, o aluno deve selecionar uma peça apresentada pelo sistema. Não tem como fazer o encaixe sem a seleção da peça.

RN13: Peças que já foram encaixadas corretamente não podem ser reposicionadas, reutilizadas ou removidas pelo aluno. O sistema deve mostrar visualmente que essa peça já foi fixada.

RN14: Se o encaixe for feito corretamente, o sistema deve exibir uma mensagem informando que a peça foi posicionada com sucesso.

RN15: Se a peça não for compatível com a área selecionada, o sistema deve impedir o encaixe e exibir uma mensagem de erro.

RN16: Após o posicionamento correto de uma peça, o sistema deve registrar o progresso do aluno.

RN10 : O sistema deve calcular a pontuação de acordo com o nível que o aluno escolheu da seguinte forma:

- FÁCIL:
    - Primeira tentativa: recebe 100% da pontuação.
    - Segunda tentativa: exibe aviso (sem perda de pontos).
    - Terceira tentativa: recebe dica e perde 20% da pontuação.
    - Quarta tentativa: ilumina o local e perde 50% da pontuação.
- MÉDIO:
    - Primeira tentativa: recebe 100% da pontuação.
    - Segunda tentativa: perde 5% da pontuação.
    - Terceira tentativa: perde 10% da pontuação.
    - Quarta tentativa: exibe aviso e perde 20% da pontuação.
    - Quinta tentativa: recebe dica e perde 30% da pontuação.
- DIFÍCIL:
    - Primeira tentativa: recebe 100% da pontuação.
    - Segunda tentativa: perde 10% da pontuação.
    - Terceira tentativa: perde 15% da pontuação.
    - Quarta tentativa: exibe aviso e perde 25% da pontuação.
    - Quinta tentativa: exibe aviso e perde 40% da pontuação

RN22: A partir do primeiro cálculo de pontuação, o sistema deve aplicar uma fórmula que visa bonificar alunos que conseguirem concluir o desafio em menor tempo, sem prejudicar os alunos que demandarem mais tempo seguindo a seguinte fórmula matemática: $X \times max(1, (2 - \frac{T}{600}))$ , onde X é a pontuação calculada de acordo com a RN10 e T é o tempo em segundos levado para concluir a montagem, de forma que o aluno ganhe um bônus se concluir em menos de 10 minutos.

## Frontend

Deve ser criada 1 página: a página de Montagem do Modo História. Deve ser seguida como base de implementação a figura abaixo.


**Figura 9**: Tela de Montagem.

O fluxo das telas deve ser o seguinte, como pode ser visualizado na Figura 9:

- **Tela de Montagem**

- Esta tela representa o início do fluxo após o aluno clicar em “Modo História”. O sistema exibe a primeira história, a área de montagem e a lista de peças disponíveis.
- O aluno deve primeiro selecionar uma peça da lista.
- Em seguida, o aluno deve clicar em uma área de encaixe no computador.
- Ao realizar esta ação, o sistema deve salvar a pontuação de acordo com a RN22.
- Com base no número de tentativas:
  - Em caso de sucesso: O sistema deve exibir uma mensagem de sucesso, atualizar a interface para mostrar a peça encaixada permanentemente, atualizar a pontuação e apresentar a próxima peça/história.
  - Em caso de falha: O sistema deve exibir a mensagem de erro informando que o encaixe é incorreto (RN15) e permitir que o aluno tente novamente.
- Após o encaixe da última peça, o sistema deve fazer uma requisição POST para o caminho “api/alunos/setPontuacao”, exibir uma mensagem de conclusão com a pontuação final e dois botões, um para retornar à “Tela de Fases” e um para ver o ranking da turma, neste caso enviando uma requisição GET para o caminho “api/salas/getRanking/{codigoUnicoDaSala}” e mostrar o ranking da turma de acordo com a resposta da requisição.

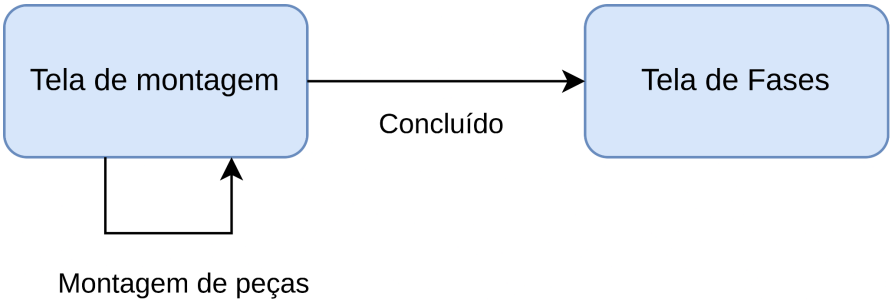


**Figura 10**: Diagrama representando o fluxo de telas de CSU08.

Espera-se que o sistema seja capaz de gerenciar o ciclo de jogo do Modo História, permitindo que o aluno interaja com as peças e a área de montagem, valide cada jogada contabilizando os pontos, realize a contagem de tempo até a última peça ser posicionada corretamente, atualize a interface visual e a pontuação de acordo com as respostas, e trate os erros de forma clara, garantindo que o progresso seja registrado corretamente.