HaLoc: Uma LocBoy hábil

Equipe de desenvolvimento

NOME	RA
João Victor Souza Lima	12822126164
Pedro Vítor da Cunha Araújo	12822123390
David Teixeira Ferreira Caldas	12822119484
Samuel Fontes da Silva	12822122655
Júlia Beatriz Silva de Lira	1282216997
Marcio Miqueias Guimarães Inácio	1282215896

Introdução

Joaquim Lisboa é proprietário de uma grande academia, com mais de 3000 alunos matriculados. Lisboa, reclama de uma falta de controle nos mais amplos setores da Locboy. Entre suas queixas estão os pagamentos atrasados que muitas vezes não são percebidos, a grande evasão de alunos depois de certa inatividade, o entendimento das despesas - bem como de um fluxo de caixa sustentável -, o controle das máquinas que estão operantes e a dificuldade de se manter a par desses acontecimentos quando viaja para o exterior. Joaquim ainda se queixa fervorosamente por não saber o quanto está ganhando. Tendo em vista suas dores, Joaquim Lisboa almeja uma solução de software que envolva monitoramento e estatísticas que ajude sua administração e funcionalidades que integrem sua academia, máquinas, treinadores e alunos.

Histórias de Usuário

História do Joaquim

Lucro

Joaquim Lisboa oferece um serviço de academia. Esse serviço funciona com o pagamento mensal dos vários planos que existem. Há recebimentos por vários meios (dinheiro, cartão de crédito e débito). No recebimento das mensalidades, que possuem uma tolerância de 5 dias excedendo o prazo de pagamento, Joaquim normalmente não consegue monitorar ao certo quanto ganha por cada meio e nem tem a noção de quem são os inadimplentes; a estes será cobrado 2% de multa e uma taxa de juros estabelecida.

Despesas

Na academia, Joaquim tem despesas de manutenção de equipamentos, insumos (luz, agua, impostos). A mão de obra da academia é: treinadores e administradores. Para os alunos da academia

ocorre um exame médico a cada três meses, esse custo reflete em um gasto de R$ 20,00 por exame ao Sr. Joaquim. Existe também de forma opcional consultas com a nutricionista que reflete um gasto específico para o aluno.

Notificação

Joaquim precisa ser notificado sobre questões de prioridade em sua academia: quando uma máquina ficar inoperante ou novamente operante, quando um pagamento estiver atrasado depois dos 5 dias de tolerância e quando um pagamento atrasado for quitado.

Administradores

Os administradores querem saber quais máquinas estão operantes e inoperantes, assim como a quantidade de vezes que ela quebrou. Eles iniciam a matrícula de cada aluno. Também gerenciam as contas dos treinadores, definem seus horários e podem permutar os alunos. Devem ter acesso aos dados de todos os alunos, assim como saber sua frequência. Os administradores devem também ter acesso aos prejuízos e lucros, assim como especificados na história do Joaquim. A administração pode entrar em contato com os treinadores e alunos.

Trabalho dos treinadores

A academia possui em média 3000 alunos cadastrados, destes, não existe ao certo uma ideia de frequência, para evitar uma possível desistência. A academia exige um monitoramento dos alunos ausentes e a notificação aos seus respectivos treinadores. Cada treinador é responsável por um grupo de alunos, ele sabe quais são esses alunos. O treinador tem o entendimento de quais exercícios seu instruído pratica, os monta com base na modalidade, e observa constantemente se seu aluno exerce com frequência X tais exercícios. O treinador possui os horários de serviço. Também é monitorado quando o comparecimento no serviço. Os instrutores podem verificar uma lista de máquinas inoperantes. O instrutor pode classificar uma máquina como operante ou inoperante.

Aluno

O aluno se matricula com um administrador, paga a primeira matrícula para assim ser efetivado. Deve lembrar de suas mensalidades e saber quando atrasadas. O aluno sabe quem é o seu instrutor. O aluno pode saber seus treinos diários.

Matrícula

A matrícula é realizada por um administrador, só é efetivada após o pagamento da primeira mensalidade, quando de fato oficializada o aluno deve fazer seus exames com o médico. Dos planos possíveis, são quatro modalidades (aeróbica, spinning, musculação e pilates). Quando o aluno é matriculado em duas modalidades, há um desconto de 10%; quando três, um desconto de 20%; quando quatro, um desconto de 30%.

Requisitos

Super administrador

A aplicação do super administrador deve possibilitar o cadastro de outros administradores

1. Um formulário na interface deve estar preparado para estruturar e enviar os dados de cadastro de administrador. O servidor deve estar preparado para receber o nome do administrador, CPF, e-mail, número de telefone, número de celular, endereço completo (rua, número, cidade, estado), CEP, valor do salário em reais e dias com os horários de serviço;
2. O sistema deve validar a veracidade do CPF, e-mail e CEP recebidos. Após, somente se os dados validados forem verídicos, deve salvar no banco de dados um administrador de nível 0 – desativado –, com sua data e hora de criação e com um identificador numérico inteiro, doravante ID, geral para todos os tipos de usuário do sistema, de 5 dígitos, incrementado pelo servidor e único;
3. Caso o cadastro ocorra com sucesso a aplicação deve ser redirecionado para a área inicial da aplicação, senão, deve permanecer na tela de cadastro e notificar o erro;
4. O sistema deve enviar ao email do administrador recém cadastrado uma URL com validade de vinte e quatro horas para a confirmação de cadastro e criação de um passe de segurança e só poderá alterar o nível do administrador para 1 – administrador intermediário –, no banco de dados, após a confirmação;
5. Caso o administrador não seja confirmado em vinte e quatro horas deverá ser permanentemente excluído do banco de dados;
6. Esta funcionalidade deve ser restrita ao usuário com um token de administrador e nível 2 – super administrador.

A aplicação do super administrador pode desativar outros administradores que estejam previamente cadastrados e ativos

1. O servidor deve estar preparado para receber o ID do administrador que deseja-se desativar;
2. Caso o ID do administrador correspondente faça parte do sistema, está ativado e não é o do super administrador, deve ser alterado o nível para 0 – desativado – no banco de dados;
3. Esta funcionalidade deve ser restrita ao usuário com um token de administrador e nível 2 – super administrador.

A aplicação do super administrador pode ativar outros administradores já cadastrados que estejam desativados

1. O servidor deve estar preparado para receber o ID do administrador que se deseja ativar;
2. Caso o ID do administrador correspondente faça parte do sistema, está cadastrado a mais de vinte e quatro horas, está desativado e se não é o do super administrador, deverá ter o nível alterado para 1 – administrador intermediário –, no banco de dados;

3. Esta funcionalidade deve ser restrita ao usuário com um token de administrador e nível 2 – super administrador.

A aplicação do super administrador pode alterar os dados de outros administradores cadastrados

1. Um formulário deve estar preparado para estruturar e enviar os dados de alteração de administrador. O servidor deve estar preparado para receber o ID do administrador que deseja-se alterar, juntamente com um ou mais dados que serão alterados: nome do administrador, e-mail, número de telefone, número de celular, endereço completo (rua, número, cidade, estado), CEP, valor do salário em reais e dias com os horários de serviço;
2. O sistema não deve aceitar alteração da ID, senha, CPF e data de criação do administrador;
3. Deve validar se o ID do administrador correspondente está cadastrado no sistema. Caso seja informado o CEP, deverá conferir se é verídico. Só poderão ser alterados os dados com esses teste sendo verídicos;
4. Caso dentre as modificações esteja um novo endereço de e-mail, o sistema deve enviar um correio eletrônico informando todos os dados alterados e esperando a confirmação para o novo e-mail, no entanto, todos os outros dados, menos o email, serão modificados sobrescrevendo os antigos no banco de dados. O email só deve ser sobrescrito no banco de dados após a confirmação do destinatário;
5. Caso o e-mail não tenha sido um dos dados modificados, o sistema deve enviar um e-mail informando os dados que foram alterados ao destinatário e os novos dados serão modificados sobrescrevendo os antigos no banco de dados;
6. Esta funcionalidade deve ser restrita ao usuário com um token de administrador e nível 2 – super administrador.

Aplicação do Administrador

Para acesso ao sistema

A aplicação de administrador deve permitir o acesso de administradores ao sistema

1. Deve ser informado o ID e a senha de um administrador ativo de qualquer nível para prosseguir;
2. O nome e a senha serão avaliados nos servidores da HaLoc conforme a correspondência real dos dados. E o servidor deve retornar ao cliente do administrador um token de acesso específico com validade de 8 horas;

A aplicação de administrador deve permitir ao administrador finalizar sua conexão antes das 8 horas possibilitadas

1. O servidor deve estar preparado para receber o ID de um administrador que deseje encerrar sua conexão;
2. Deve então excluir um token correspondente ao ID na lista de tokens ativos dos administradores do sistema;

3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação de administrador deve exibir todos os administradores do sistema

1. Uma interface deve estar preparada para solicitar ao servidor e apresentar em uma lista com cada administrador: o ID, nome, nível e mostrar mais. Também deve apresentar a opção de cadastrar um administrador;
2. Mostrar mais deve exibir o CPF, e-mail, número de telefone, número de celular, endereço completo com CEP, valor do salário em reais e dias com os horários de serviço na mesma página, apenas estendido abaixo da linha original e tendo agora uma opção de mostrar menos para esta linha;
3. Caso o administrador tenha nível 2 – super administrador –, na exibição de nível poderá ser editado o status de ativo ou inativo (0 ou 1) e ter a opção de ir para edição dos dados;
4. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador deve permitir alterar seus próprios dados

1. Um formulário na aplicação deve estar preparado para estruturar e enviar os dados de alteração do administrador. O servidor deve estar preparado para receber o ID do próprio administrador que deseja alterar, juntamente com um ou mais dados que serão alterados: nome do administrador, e-mail, número de telefone, número de celular, endereço completo (rua, número, cidade, estado) e CEP;
2. O sistema não deve aceitar alteração da ID, senha, CPF e data de criação do treinador;
3. Caso seja informado o CEP só poderão ser alterados os dados caso ele seja válido;
4. Caso dentre as modificações esteja um novo endereço de e-mail, o sistema deve enviar um correio eletrônico informando todos os dados alterados e esperando a confirmação para o novo e-mail, no entanto, todos os outros dados, menos o email, serão modificados sobrescrevendo os antigos no banco de dados. O email só deve ser sobrescrito no banco de dados após a confirmação;
5. Caso o e-mail não tenha sido um dos dados modificados, o sistema deve enviar um e-mail informando os dados que foram alterados e os novos dados serão modificados sobrescrevendo os antigos;
6. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível e que o ID coincida com o que está sendo alterado.

A aplicação do administrador deve possuir uma página de notificações

1. As notificações devem ser guardadas no banco com título, data, hora, descrição, para qual nível de administrador podem ser exibidas e ID auto incrementado;
2. Devem ser exibidas em uma página exclusiva para notificações que exibirá o título, descrição, data e hora;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

O sistema RFID deve monitorar a presença de administradores, treinadores e alunos na academia

1. Caso o receptor geral da academia identifique a presença do usuário, emitindo seu ID em uma distância pré-determinada, o dispositivo deve informar para o servidor salvar no banco de dados em um log de frequência a data, o horário, RFID, um ID específico para log e o horário de entrada;
2. Caso o receptor geral da academia, que antes identificava a presença do RFID do funcionário, perceba que ele não está mais na distância determinada, deve esperar um tempo pseudo aleatório entre 1 minuto e 5 minutos para então informar para o servidor salvar no banco de dados em um log de frequência a data, horário, RFID, um ID específico para o log e momento de saída.

A aplicação de administrador deve monitorar irregularidades de funcionários

1. O sistema, utilizando os logs de frequência e horários de serviço dos treinadores e administradores deve estimar se houve algum atraso ou não-comparecimento no trabalho e criar um log para o mesmo;
2. Caso o treinador saia do seu horário de serviço múltiplas vezes o sistema deve gerar uma irregularidade para cada não-comparecimento;
3. As faltas serão exibidas de forma organizada na aplicação e poderão ser justificadas com a possibilidade de opções pré-estabelecidas: troca de turno combinada, atestado médico ou folga;
4. Caso o administrador que esteja acessando a página tenha nível 2, deverá ver e poderá justificar as faltas de outros administradores;
5. Qualquer falta de 15 minutos dos treinadores deve ocasionar uma notificação sobre a mesma para os administradores. Para o administrador de nível 2 deve também ser notificado falta de 15 minutos de administradores;
6. Caso haja uma justificativa plausível para o atraso do funcionário, pode ser feita adicionando uma descrição detalhando a situação. Assim será regularizada;
7. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação de administrador pode adicionar hora extra aos funcionários

1. Deve ser informado o dia da hora extra juntamente com a hora de início e fim;
2. Deve ser confirmado pelo sistema o comparecimento na hora extra;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

O administrador deve marcar seu horário de início e o de saída do período de serviço

1. Deve ser informado o horário que entrou na academia para comprimento do ofício;
2. Deve ser informado o horário de finalização das atividades no momento de saída;
3. Deve possuir uma interface receptora RFID apropriada para a marcação de turnos de funcionários que receba o ID do administrador.

Administrador para treinadores

A aplicação de administrador deve ter uma visão de todos os treinadores do sistema

1. A aplicação deve estar preparada para solicitar ao servidor e apresentar em uma lista com cada treinador: o ID, nome, status – que na mesma tela pode ser alterado –, a opção de ir para edição de dados e mostrar mais. Também deve apresentar a opção de cadastrar um treinador;
2. Mostrar mais deve exibir o CPF, e-mail, número de telefone, número de celular, endereço completo com CEP, valor do salário em reais e dias com os horários de serviço na mesma página apenas estendendo abaixo da linha original e tendo agora uma opção de mostrar menos para esta linha;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação de administrador pode ter uma visão mais detalhada de um treinador no sistema

1. Deve aparecer em cabeçalho o ID do treinador, nome, status, CPF, e-mail, número de telefone, número de celular, endereço completo com CEP, salário e dias com horários de serviço;
2. Deve ter uma listagem de faltas e atrasos no serviço;
3. Deve ter uma listagem de alunos que estão sobre a instrução de tal treinador;
4. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação de administrador deve permitir que sejam cadastrados treinadores

1. Um formulário no aplicação deve estar preparado para estruturar e enviar os dados. O servidor deve estar preparado para receber o nome do treinador, CPF, e-mail, número de telefone, número de celular, endereço completo (rua, número, cidade, estado), CEP, valor do salário em reais e dias com os horários de serviço;
2. O sistema deve validar a veracidade do CPF, e-mail e CEP recebidos. Após, somente se os dados validados forem verídicos, deve salvar no banco de dados um treinador de status 0 – desativado –, com sua data e hora de criação e com um ID, geral para todos os tipos de usuário do sistema, de 5 dígitos, incrementado pelo servidor e único;
3. Caso o cadastro ocorra com sucesso a aplicação deve ser redirecionada para a área inicial, senão, deve permanecer na tela de cadastro e notificar o erro;
4. O sistema deve enviar ao email do treinador recém cadastrado uma URL com validade de vinte e quatro horas para a confirmação de cadastro e criação de um passe de segurança e só poderá alterar o status do treinador para 1 – ativado –, no banco de dados, após a confirmação;
5. Caso o treinador não seja confirmado em vinte e quatro horas deverá ser permanentemente excluído do banco de dados;
6. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador deve permitir desativar treinadores que estejam previamente cadastrados e ativos

1. O servidor deve estar preparado para receber o ID do treinador que se deseja desativar;
2. Caso o ID do treinador correspondente faça parte do sistema e está ativo, deve ser alterado o status para 0 – desativado – no banco de dados;
3. Esta funcionalidade deve ser restrita a um usuário com um token de administrador em qualquer nível.

A aplicação do administrador pode ativar treinadores já cadastrados que estejam desativados

1. O servidor deve estar preparado para receber o ID do treinador que se deseja ativar;
2. Caso o ID do treinador correspondente faça parte do sistema e está cadastrado a mais de vinte e quatro horas, deverá ter o status do treinador alterado para 1 – ativado –, no banco de dados;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode alterar os dados de treinadores cadastrados

1. Um formulário na interface com o administrador deve estar preparado para estruturar e enviar os dados de alteração de treinador. O servidor deve estar preparado para receber o ID do treinador que deseja-se alterar, juntamente com um ou mais dados que serão alterados: nome do treinador, e-mail, número de telefone, número de celular, endereço completo (rua, número, cidade, estado), CEP, valor do salário em reais e dias com os horários de serviço;
2. O sistema não deve aceitar alteração da ID, senha, CPF e data de criação do treinador;
3. Deve validar se o ID do treinador correspondente está cadastrado no sistema. Caso seja informado o CEP, deverá conferir se é verídico. Só poderão ser alterados os dados com esses teste sendo verídicos;
4. Caso dentre as modificações esteja um novo endereço de e-mail, o sistema deve enviar um correio eletrônico informando todos os dados alterados e esperando a confirmação para o novo e-mail, no entanto, todos os outros dados, menos o email, serão modificados sobrescrevendo os antigos no banco de dados. O email só deve ser sobrescrito no banco de dados após a confirmação do destinatário;
5. Caso o e-mail não tenha sido um dos dados modificados, o sistema deve enviar um e-mail informando os dados que foram alterados ao destinatário e os novos dados serão modificados sobrescrevendo os antigos no banco de dados;
6. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode ter uma visão de todas as máquinas da academia

1. Deve conseguir ver as máquinas que estão operantes em um tópico “máquinas operantes” e a quantidade de vezes que a mesma quebrou;

2. Deve conseguir ver as máquinas que estão inoperantes em um tópico “máquinas inoperantes”, juntamente com a data que ficou inoperante;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

Administrador para alunos

A aplicação de administrador deve possuir uma visualização de todos os alunos do sistema

1. A aplicação deve estar preparada para solicitar ao servidor e apresentar em uma lista com cada aluno mostrando o ID, nome, status – que na mesma tela pode ser alterado –, a opção de ir para edição de dados e a de mostrar mais. Também deve apresentar a opção de cadastrar um aluno;
2. “Mostrar mais” deve exibir o CPF, e-mail, número de telefone, número de celular, endereço completo com CEP, treinador responsável, valor da mensalidade e modalidades que participa na mesma página apenas estendendo abaixo da linha original e tendo agora uma opção de mostrar menos para esta linha;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação de administrador pode ver de forma mais detalhada e centralizada as informações sobre o aluno

1. Na visão detalhada das informações do aluno deve aparecer em cabeçalho o ID, nome, status, CPF, e-mail, número de telefone, número de celular, endereço completo com CEP, valor da mensalidade, modalidades e treinador responsável;
2. Deve ser exibida uma listagem com as mensalidades do aluno – ordem decrescente por data – que podem ser filtradas em todas, quitadas, pendentes e atrasadas;
3. Deve puder conferir os relatórios gerados por um treinador para o aluno;
4. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível;

A aplicação de administrador deve permitir que sejam matriculados alunos

1. Um formulário deve estar preparado para estruturar e enviar os dados. O servidor deve estar preparado para receber o nome do aluno, CPF, e-mail, número de telefone, número de celular, endereço completo (rua, número, cidade, estado), CEP, modalidades que deseja participar;
2. O sistema deve validar a veracidade do CPF, e-mail e CEP recebidos. Após, somente se os dados validados forem verídicos, deve salvar no banco de dados um aluno de status 0 – desativado –, mensalidade com base em suas modalidades, com sua data e hora de criação e com um ID, geral para todos os tipos de usuário do sistema, de 5 dígitos, incrementado pelo servidor e único;
3. Caso o cadastro ocorra com sucesso a aplicação deve ser redirecionado para seu início, senão, deve permanecer nocadastro e notificar o erro;

4. O sistema deve enviar ao email do aluno recém cadastrado uma URL com validade única para a confirmação de cadastro e criação de um passe de segurança;
5. O sistema só deve alterar o status do aluno para 1 – ativado –, no banco de dados, após o pagamento da primeira mensalidade e ele deve ser atribuído a um treinador;
6. Caso o pagamento não seja confirmado em 3 dias úteis o aluno deve ser excluído do sistema;
7. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador deve permitir desativar alunos que estejam previamente cadastrados e ativos

1. O servidor deve estar preparado para receber o ID do aluno que se deseja desativar;
2. Caso o ID do aluno correspondente faça parte do sistema e está ativo, deve ser alterado o status para 0 – desativado – no banco de dados;
3. Esta funcionalidade deve ser restrita a um usuário com um token de administrador em qualquer nível.

A aplicação do administrador pode ativar alunos já cadastrados que estejam desativados

1. O servidor deve estar preparado para receber o ID do aluno que se deseja ativar;
2. Caso o ID do aluno correspondente faça parte do sistema e está cadastrado a mais de três dias, deverá ter o status do treinador alterado para 1 – ativado –, no banco de dados;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode alterar os dados de alunos cadastrados

1. Um formulário na aplicação deve estar preparado para estruturar e enviar os dados de alteração de aluno. O servidor deve estar preparado para receber o ID do aluno que deseja-se alterar, juntamente com um ou mais dados que serão alterados: nome do aluno, e-mail, número de telefone, número de celular, endereço completo (rua, número, cidade, estado), CEP e modalidades;
2. O sistema não deve aceitar alteração da ID, senha, CPF e data de criação do aluno;
3. Deve validar se o ID do aluno correspondente está cadastrado no sistema. Caso seja informado o CEP, deverá conferir se é verídico. Só poderão ser alterados os dados com esses teste sendo verídicos;
4. Caso dentre as modificações esteja um novo endereço de e-mail, o sistema deve enviar um correio eletrônico informando todos os dados alterados e esperando a confirmação para o novo e-mail, no entanto, todos os outros dados, menos o e-mail, devem ser alterados no banco de dados. O email só deve ser sobrescrito após a confirmação do destinatário;
5. Caso o e-mail não tenha sido um dos dados modificados, o sistema deve enviar um e-mail informando os dados que foram alterados ao destinatário e os novos dados sobrescrevem os antigos;
6. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode validar a alteração de dados solicitada pelo aluno ou treinador

1. Devem aparecer pedidos de alteração de dados de alunos ou treinadores para administradores em uma interface de pedidos de alteração de dados;
2. O pedido deve ser aceito ou rejeitado totalmente com base nos critérios do avaliador;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode alterar um treinador de um aluno

1. Deve ver quem é o treinador atual do aluno;
2. Deve conseguir em uma listagem de treinadores, na hora da mudança, ver com quantos alunos os referentes já estão;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

Administrador para finanças

A aplicação do administrador deve ter uma visão de todas as mensalidades na academia

1. Deve ser exibida uma listagem que pode ser filtrada em todas, quitadas, pendentes e atrasadas;
2. Deve ser filtrada em todos os alunos ou aluno específico;
3. Deve ser exibido um panorama de valor total para quitadas, pendentes e atrasadas;
4. As mensalidades pendentes devem ser as referentes ao mês corrente;
5. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode ter uma visão de todos os caixas mensais da academia

1. Deve poder restringir os caixas por um período de tempo: geral, 5 anos até 1 anos ou os últimos 6 ou 3 meses;
2. Cada caixa deve estar aberto ou fechado e isso deve ser definido pelo administrador;
3. A cada primeiro dia de um mês é aberto um novo caixa automáticamente;
4. Deve ser exibido um panorama geral do saldo do período;
5. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode ter uma visão específica para um caixa mensal

1. Deve mostrar as despesas cadastradas e o último administrador que executou alguma alteração nela;
2. Deve mostrar as despesas médicas do mês, bem como os alunos referentes;
3. Deve mostrar um panorama dos gastos do mês, apurado e saldo;

4. Deve mostrar “x” maiores gastos do mês, no qual “x” é definido pelo administrador;
5. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode cadastrar uma despesa em um caixa

1. O caixa que receberá a despesa deve estar aberto;
2. A despesa deve ter valor, título e pode ter descrição;
3. Deve ser definido se é uma despesa pontual para o mês ou se há uma periodicidade: se repete todos os meses ou por N meses;
4. Deve ser salva a data e hora do cadastro da despesa e o administrador que a cadastrou;
5. Caso tenha uma periodicidade deve ser definido se o valor da despesa é constante para todos os meses;
6. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode excluir uma despesa cadastrada em um caixa

1. Caso haja uma periodicidade indefinida deve ser informado se a exclusão é apenas para o caixa atual ou se refletirá para os próximos caixas que serão abertos;
2. Não deve ser excluído automaticamente as despesas posteriores que já foram criadas, só não ocorrerá mais para as novas despesas que serão abertas em novos meses pelo sistema;
3. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

A aplicação do administrador pode editar uma despesa cadastrada em um caixa

1. A despesa pode ter o nome, título ou descrição alterados;
2. Deve ser estabelecido se a modificação ocorrerá para os próximos caixas que serão gerados;
3. Não deve ser alterado automaticamente as despesas posteriores já cadastradas;
4. Esta funcionalidade deve ser restrita a usuários com um token de administrador em qualquer nível.

Aplicação de treinador

A aplicação de treinador deve permitir o acesso de treinadores ao sistema

1. Deve ser informado o ID e a senha de um treinador ativo para prosseguir;
2. O ID e a senha serão avaliados nos servidores da HaLoc conforme a correspondência real dos dados. E o servidor deve retornar a aplicação do treinador um token de acesso específico para treinador.

A aplicação de treinador deve permitir ao treinador finalizar sua conexão

1. O servidor deve estar preparado para receber o ID de um treinador que deseje encerrar sua conexão;

2. Deve então excluir um token correspondente ao ID na lista de tokens ativos dos treinadores do sistema;
3. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

A aplicação de treinador deve possuir uma visualização de todos os seus alunos

1. A aplicação deve solicitar ao servidor e apresentar em uma lista com cada aluno ativo mostrando o ID, nome, modalidades e a opção de mostrar mais;
2. “Mostrar mais” deve exibir o e-mail, número de telefone, número de celular e modalidades que participa na mesma página apenas estendendo abaixo da linha original e tendo agora uma opção de mostrar menos para esta linha;
3. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

A aplicação de treinador pode ver de forma mais detalhada e centralizada as informações sobre seu instruído

1. Na visão detalhada das informações do aluno deve aparecer em cabeçalho o ID, nome, e-mail, número de telefone, número de celular, modalidades e treinos;
2. Pode conferir uma frequência mensal do aluno em visitações na academia;
3. Deve listar os exames médicos e suas datas;
4. Deve puder conferir os relatórios gerados para o aluno e ir para a edição do mesmo;
5. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

A aplicação de treinador deve gerar um relatório mensal para cada instruído

1. Automáticamente no dia 31 do mês deve ser gerado um relatório em branco para cada aluno com a autoria do treinador responsável;
2. O treinador deve escrever neste arquivo noções sobre a frequência do aluno, progresso e saúde;
3. Caso o aluno tenha sido transferido de outro treinador, seus relatórios antigos acompanham, mas não podem ser editados pelo novo treinador;
4. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

A aplicação de treinador deve ter a capacidade de montar os treinos do aluno

1. O treinador deve montar o treino de seu instruído com base nas modalidades que participa;
2. No caso de uma redefinição nos treinos o aluno deve ser notificado por e-mail;
3. Caso o aluno venha transferido de outro treinador, o treino anteriormente montado deve acompanhá-lo;
4. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

A aplicação do treinador deve validar solicitação para o nutricionista de aluno

1. A aplicação deve receber pedidos para marcar o nutricionista, com a data escolhida, e o treinador terá que verificar a possibilidade para então deferir ou indeferir o pedido;
2. O aluno deve ter o conhecimento sobre o resultado da validação;
3. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

A aplicação do treinador pode ter uma visão de todas as máquinas da academia

1. Deve conseguir ver as máquinas que estão operantes em um tópico “máquinas operantes” e a quantidade de vezes que a mesma quebrou;
2. Deve conseguir ver as máquinas que estão inoperantes em um tópico “máquinas inoperantes”, juntamente com a data que ficou inoperante;
3. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

A aplicação do treinador pode classificar uma máquina da academia como inoperante

1. Deve se tratar de uma máquina que esteja previamente operante;
2. O treinador deve emitir seu ID próximo da máquina após solicitar a mudança de estado.
3. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

A aplicação do treinador pode classificar uma máquina da academia como concertada

1. Deve se tratar de uma máquina que esteja previamente inoperante;
2. O treinador deve emitir seu ID próximo da máquina após solicitar a mudança de estado;
3. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

A aplicação de treinador deve solicitar ao aluno uma data para o exame médico

1. Deve ser permitido após três meses do último exame;
2. Deve ser solicitado outra data caso o aluno não concorde com a proposta;
3. Esta funcionalidade deve ser restrita a usuários com um token de treinador.

O treinador deve marcar seu horário de início e o de saída do período de serviço

4. Deve ser informado o horário que entrou na academia para comprimento do ofício;
5. Deve ser informado o horário de finalização das atividades no momento de saída;
6. Deve possuir uma interface receptora RFID apropriada para a marcação de turnos de funcionários que receba o ID do treinador.

Aplicação de aluno

A aplicação de aluno deve permitir o acesso de alunos ao sistema

3. Deve ser informado o ID e a senha de um aluno ativo para prosseguir;
4. O ID e a senha serão avaliados nos servidores da HaLoc conforme a correspondência real dos dados. E o servidor deve retornar a aplicação do aluno um token de acesso específico para aluno.

A aplicação de aluno deve permitir ao aluno finalizar sua conexão

4. O servidor deve estar preparado para receber o ID de um aluno que deseje encerrar sua conexão;

5. Deve então excluir um token correspondente ao ID na lista de tokens ativos dos alunos do sistema;
6. Esta funcionalidade deve ser restrita a usuários com um token de aluno.

O aluno deve ter uma mensalidade com base em algumas regras de negócio

1. Deve haver mensalidade apenas alunos com status ativo;
2. A mensalidade deve ser correspondente ao valor das modalidades que está cadastrado;
3. Caso o aluno esteja em duas modalidades haverá um desconto de 10% no total;
4. Caso o aluno esteja em três modalidades haverá um desconto de 20% no total;
5. Caso o aluno esteja em quatro modalidades haverá um desconto de 30% no total.

O aluno deve ter um panorama de suas mensalidades

1. Deve ser notificado sobre mensalidades atrasadas;
2. Deve ser notificado sobre mensalidades com data de vencimento próxima;
3. Esta funcionalidade deve ser restrita a usuários com um token de aluno.

O aluno deve pagar suas mensalidades

1. Deve puder optar por PIX, cartão ou dinheiro. As duas primeiras opções podem ser feitas pela aplicação do aluno;
2. Esta funcionalidade deve ser restrita a usuários com um token de aluno.

A aplicação do aluno deve possibilitar uma visão das informações de seu treinador

1. Deve ser exibido o nome do instrutor com destaque;
2. Deve mostrar os horários e dias de serviço, e-mail e celular;
3. Esta funcionalidade deve ser restrita a usuários com um token de aluno.

A aplicação do aluno pode validar a data do exame médico escolhida pelo treinador

1. O aluno deve aceitar ou não a data com seus critérios de disponibilidade;
2. Caso não possa na data informada, deverá receber uma nova proposta do treinador;
3. Esta funcionalidade deve ser restrita a usuários com um token de aluno.

A aplicação do aluno pode marcar horário com o nutricionista

1. O aluno deve selecionar uma data que deseje comparecer ao nutricionista;
2. O pedido deve ser ou não deferido;
3. Esta funcionalidade deve ser restrita a usuários com um token de aluno.

A aplicação do aluno deve possibilitar a visão dos treinos do aluno

1. Deve mostrar o que deve ser feito durante a semana;
2. Deve mostrar se está sendo comprido;
3. Esta funcionalidade deve ser restrita a usuários com um token de aluno.

Requisitos não-funcionais

O sistema deve, em uma arquitetura cliente-servidor, possuir um algoritmo de Criptografia TLS.

1. Deve possuir uma chave pública e uma privada que garantam a criptografia dos dados do cliente ao servidor;
2. Deve ser reconhecido como aplicação segura por navegadores oficiais;
3. Deve possuir um serviço com total disponibilidade para todas as aplicações do sistema.

As aplicações devem aproveitar a maior base de código possível

1. Deve ser utilizadas tecnologias semelhantes para ambas aplicações;
2. As tecnologias devem ser de renderização dinâmica.

Os dados devem ser exibidos aproveitando os melhores conceitos de Power BI

1. Deve haver visão estática dos dados por gráficos bem elaborados;
2. Técnicas de snapshots para melhor entendimento dos dados por períodos.

Interface de comunicação API REST FULL

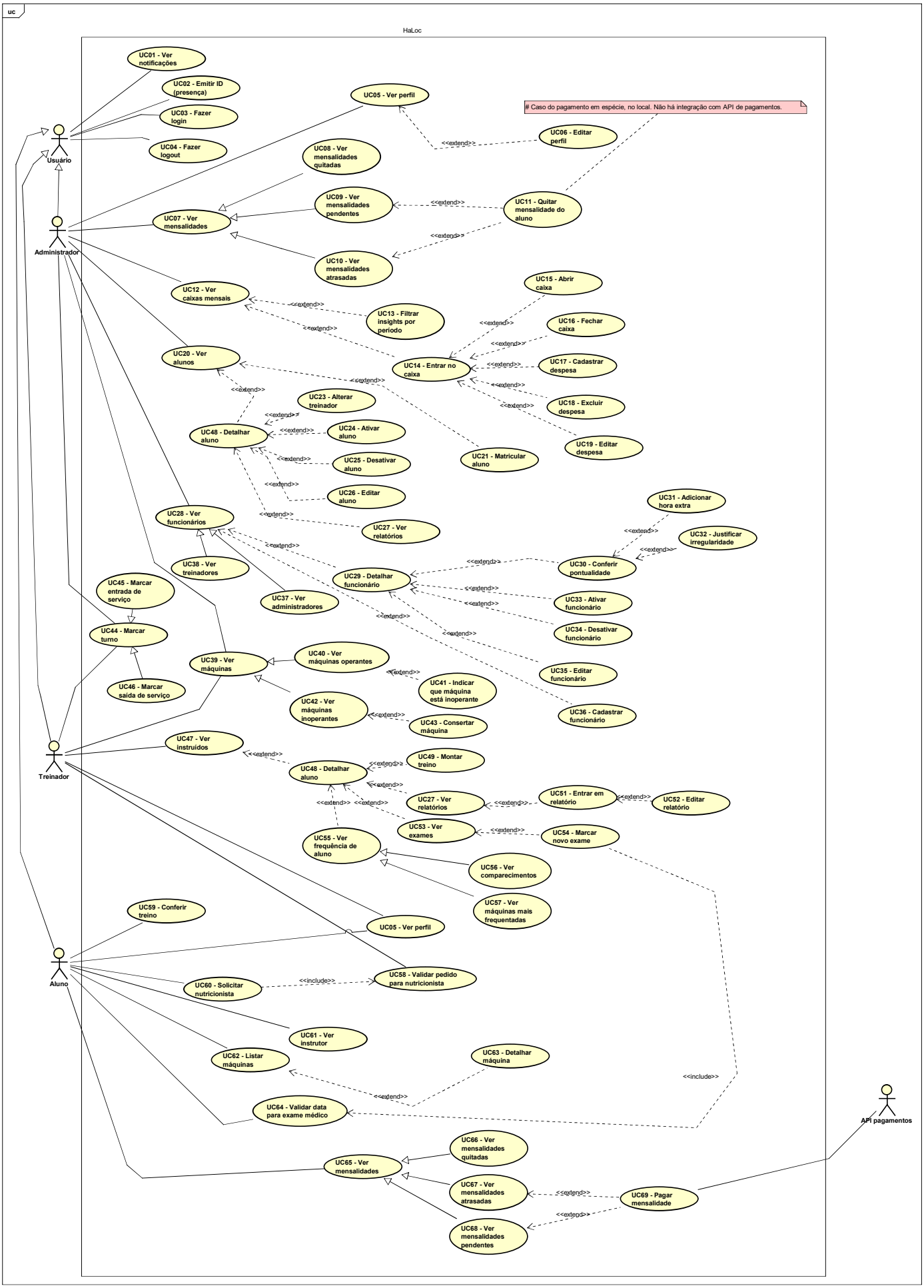
1. Os endpoints devem ser objetivos;
2. Os métodos HTTP devem ser utilizados de forma coerente.

Deve haver a transcrição de elementos não-verbais

1. Técnicas padrões das linguagens de marcação devem ser aproveitadas;
2. Recursos de leitura de aparelhos e software devem ser aproveitados.

Cada funcionalidade deve estar dentro de seu escopo

1. Deve evitar a poluição visual nas aplicações;
2. Cada função estará apenas na sua aplicação específica;
3. Deve haver aplicações diferentes para o administrador, treinador e aluno.


uc
HaLoc
UC01 - Ver notificações
UC02 - Emitir ID (presença)
UC03 - Fazer login
UC04 - Fazer logout
UC05 - Ver perfil
UC06 - Editar perfil
UC07 - Ver mensalidades
UC08 - Ver mensalidades quitadas
UC09 - Ver mensalidades pendentes
UC10 - Ver mensalidades atrasadas
UC11 - Quitar mensalidade do aluno
UC12 - Ver caixas mensais
UC13 - Filtrar insights por período
UC14 - Entrar no caixa
UC15 - Abrir caixa
UC16 - Fechar caixa
UC17 - Cadastrar despesa
UC18 - Excluir despesa
UC19 - Editar despesa
UC20 - Ver alunos
UC21 - Matricular aluno
UC23 - Alterar treinador
UC24 - Ativar aluno
UC25 - Desativar aluno
UC26 - Editar aluno
UC27 - Ver relatórios
UC28 - Ver funcionários
UC29 - Detalhar funcionário
UC30 - Conferir pontualidade
UC31 - Adicionar hora extra
UC32 - Justificar irregularidade
UC33 - Ativar funcionário
UC34 - Desativar funcionário
UC35 - Editar funcionário
UC36 - Cadastrar funcionário
UC37 - Ver administradores
UC38 - Ver treinadores
UC39 - Ver máquinas
UC40 - Ver máquinas operantes
UC41 - Indicar que máquina está inoperante
UC42 - Ver máquinas inoperantes
UC43 - Consertar máquina
UC44 - Marcar turno
UC45 - Marcar entrada de serviço
UC46 - Marcar saída de serviço
UC47 - Ver instruídos
UC48 - Detalhar aluno
UC49 - Montar treino
UC51 - Entrar em relatório
UC52 - Editar relatório
UC53 - Ver exames
UC54 - Marcar novo exame
UC55 - Ver frequência de aluno
UC56 - Ver comparecimentos
UC57 - Ver máquinas mais frequentadas
UC58 - Validar pedido para nutricionista
UC59 - Conferir treino
UC60 - Solicitar nutricionista
UC61 - Ver instrutor
UC62 - Listar máquinas
UC63 - Detalhar máquina
UC64 - Validar data para exame médico
UC65 - Ver mensalidades
UC66 - Ver mensalidades quitadas
UC67 - Ver mensalidades atrasadas
UC68 - Ver mensalidades pendentes
UC69 - Pagar mensalidade
Caso do pagamento em espécie, no local. Não há integração com API de pagamentos.
Usuário
Administrador
Treinador
Aluno
API pagamentos
<<extend>>
<<include>>

Detalhamento Caso de Uso

Primeiro Detalhamento de Caso de Uso

Nome: UC48 - Detalhar aluno

Atores

Administrador

Visão geral

Um administrador autenticado detalha os dados de um aluno de sua escolha.

Outros casos de uso relacionados:

UC20 - Ver alunos, UC23 - Alterar treinador, UC24 - Ativar aluno, UC25 - Desativar aluno, UC26 - Editar aluno e UC27 - Ver relatórios.

Cenário típico

1. Um usuário autenticado administrador de qualquer nível (1 ou 2) envia uma requisição com o ID do aluno que deseja ver detalhadamente;
2. O sistema encontra o aluno especificado pelo ID;
3. O sistema exibe de forma organizada as seguintes informações: ID, nome, status, CPF, e-mail, número de telefone, número de celular, endereço completo com CEP, valor da mensalidade, modalidades e treinador responsável;
4. O sistema exibe as mensalidades do aluno com prioridade para as atrasadas, destacadas em vermelho. Enquanto as pendentes estarão na cor amarela e as pagas verde;
5. O sistema exibe todos os relatórios descritos pelo treinador para o aluno;
6. O caso de uso termina com sucesso.

Cenários alternativos

2.a ID não existente

1. O sistema não reconhece nenhum aluno com o ID informado para que seja especificado;
2. O sistema retorna para o administrador o status HTTP 404 (Page not Found);
3. A aplicação exibe uma tela para o usuário informando a raiz do problema: não aluno há aluno matriculado para este ID, você quis dizer (ID com numeração próxima);
4. O caso de uso é abortado.

4.a Aluno inativo

1. O sistema verifica que aluno está com o status igual a False (inativo) e que não há mais mensalidades pendentes, apenas pagas ou atrasadas;

2. O sistema exibe as mensalidades quitadas, as atrasadas destacada e por último diz que o aluno não está mais ativo na academia, portanto, não há mensalidades futuras;
3. O caso de uso volta ao cenário típico no passo 5.

Pré-condições

1. Antes de iniciar o usuário deve ser um administrador logado no sistema;
2. Um ID numérico inteiro é passado como argumento.

Pós-condição

1. O administrador deve ter a visualização detalhada dos dados do aluno solicitado pelo ID.

Segundo Detalhamento de Caso de Uso

Nome: UC43 - Consertar máquina

Atores

Treinador

Visão geral

Um treinador autenticado indica para o sistema que uma máquina volta ao pleno funcionamento.

Outros casos de uso relacionados:

UC39 - Ver máquinas, UC42 - Ver máquinas inoperantes

Cenário típico

1. Um usuário autenticado treinador informa o código de uma máquina inoperante que pretende ser consertada;
2. O sistema identifica a máquina inoperante e espera para que o treinador confirme aproximando seu ID da máquina concertada;
3. O treinador envia seu ID para a máquina confirmado a efetivação do processo;
4. O sistema recebe o ID do treinador como confirmação e muda a situação da máquina para operante (True) ao passo que atribui o treinador como responsável pela operação de concerto e incrementa em um as vezes que a máquina ficou inoperante;
5. O caso de uso termina com sucesso.

Cenários alternativos

2.a O código da máquina não existe

1. O sistema não reconhece nenhuma máquina com o código informado;

2. O sistema retorna para o treinador o status HTTP 400 (Bad request), indicando que a máquina informada não se trata de uma opção válida;
3. A aplicação exibe uma tela para o treinador informando a raiz do problema: não há máquina na academia com o determinado código, você quis dizer (código com numeração próxima);
4. O caso de uso é abortado.

2.b A máquina já está com status operante

1. O sistema reconhece que a máquina não precisa ser consertada pois já está com a situação operante (True);
2. O sistema retorna para o treinador o status HTTP 423 (Locked), indicando que a operação foi bloqueada;
3. A aplicação exibe uma tela para o treinador informando a raiz do problema: a máquina informada já está operante, tente informar que ela foi danificada primeiro;
4. O caso de uso é abortado.

4.a O ID pertence e um treinador inativo

1. O sistema não reconhece o ID informado como a de um treinador ativo;
2. O sistema retorna para o treinador o status HTTP 423 (Locked), indicando que a operação foi bloqueada;
3. A aplicação exibe uma tela para o treinador informando a raiz do problema: não é possível para um treinador com ID inativo mudar a situação dos aparelhos;
4. O caso de uso é abortado.

4.b O ID informado não existe no sistema

1. O sistema não reconhece o ID informado como a de um treinador cadastrado;
2. O sistema retorna para o treinador que fez a requisição o status HTTP 400 (Bad request), indicando que o ID informado está errado;
3. A aplicação exibe uma tela para o treinador informando a raiz do problema: não é possível para um treinador com ID que não existe no sistema alterar a situação dos aparelhos;
4. O caso de uso é abortado.

4.c O ID informado não condiz com o do treinador que fez a primeira requisição de concerto

1. O sistema verifica que o ID informado na operação de confirmação não condiz com o ID que operou o concerto;
2. O sistema retorna para o treinador que fez a requisição o status HTTP 423 (Locked), indicando que o ID informado é inválido;
3. A aplicação exibe uma tela para o treinador informando a raiz do problema: não é possível que a confirmação da operação seja efetuada com um ID diferente do qual iniciou o concerto;
4. O caso de uso é abortado.

Pré-condições

1. Antes de iniciar o usuário deve ser um treinador logado no sistema.

Pós-condição

1. A máquina passa a ter uma situação operante, o treinador deve ser alocado como a última pessoa que efetuou um concerto na máquina - objeto das requisições - e as vezes que ela ficou inoperante será incrementada em um.

Terceiro Detalhamento de Caso de Uso

Nome: UC69 - Pagar mensalidade

Atores

Aluno e API de Pagamentos

Visão geral

Um aluno autenticado no sistema paga uma mensalidade que esteja disponível.

Outros casos de uso relacionados:

UC68 - Ver mensalidades pendentes, UC67 - Ver mensalidades atrasadas e UC65 - Ver mensalidades.

Cenário típico

1. Um usuário autenticado como aluno faz uma requisição para o sistema informando seu próprio ID, meio de pagamento e a chave da mensalidade;
2. O sistema recebe os dados da requisição e cruza-os para então dar continuidade;
3. O sistema utiliza uma API de pagamentos para dar continuidade para o processo dependendo do meio de pagamento (PIX, cartão de crédito ou débito);
4. Caso o pagamento seja PIX, o sistema deve receber um QR Code referente ao pagamento e exibir ao aluno;
5. o sistema recebe a confirmação da API, muda o status para paga (True) e sinaliza que está regular (True);
6. O sistema notifica aos administradores que uma mensalidade foi paga;
7. Caso a mensalidade paga estivesse anteriormente irregular (False), o sistema notifica ao administrador que o aluno X quitou uma mensalidade atrasada;
8. O caso de uso termina com sucesso.

Cenários alternativos

2.a O ID do aluno não corresponde ao responsável da mensalidade

1. O sistema reconhece que o ID passado junto a mensalidade não é referente ao ID responsável a pagar a mensalidade;
2. O sistema retorna para o aluno o status HTTP 423 (Locked);
3. A aplicação do usuário informa o problema dizendo que o aluno não é autorizado a inferir ações sobre a mensalidade;
4. O caso de uso é abortado.

2.b A mensalidade já está quitada

1. O sistema reconhece que a mensalidade referente já está com status de paga (True);
2. O sistema retorna para o aluno o status HTTP 423 (Locked);
3. A aplicação do usuário informa o problema dizendo que o aluno não é autorizado a inferir ações sobre mensalidades já quitadas;
4. O caso de uso é abortado.

2.c A mensalidade não foi encontrada

1. O sistema não identifica nenhuma mensalidade correspondente a hash informada;
2. O sistema retorna para o aluno o status HTTP 400 (Bad request);
3. A aplicação do usuário informa o problema dizendo que o aluno informou uma mensalidade não existente;
4. O caso de uso é abortado.

6.a A API de pagamentos não conseguiu efetuar o processamento

1. O sistema não recebe uma resposta válida sobre a confirmação da operação;
2. O sistema retorna para o aluno o status HTTP 502 (Bad Gateway);
3. A aplicação do usuário informa o problema dizendo que ocorreu um problema com o sistema de pagamentos;
4. O caso de uso é abortado.

6.a A API de pagamentos pode efetuar o processamento, mas o meio de pagamento do usuário não é condizente com dados reais ou proprietários do mesmo

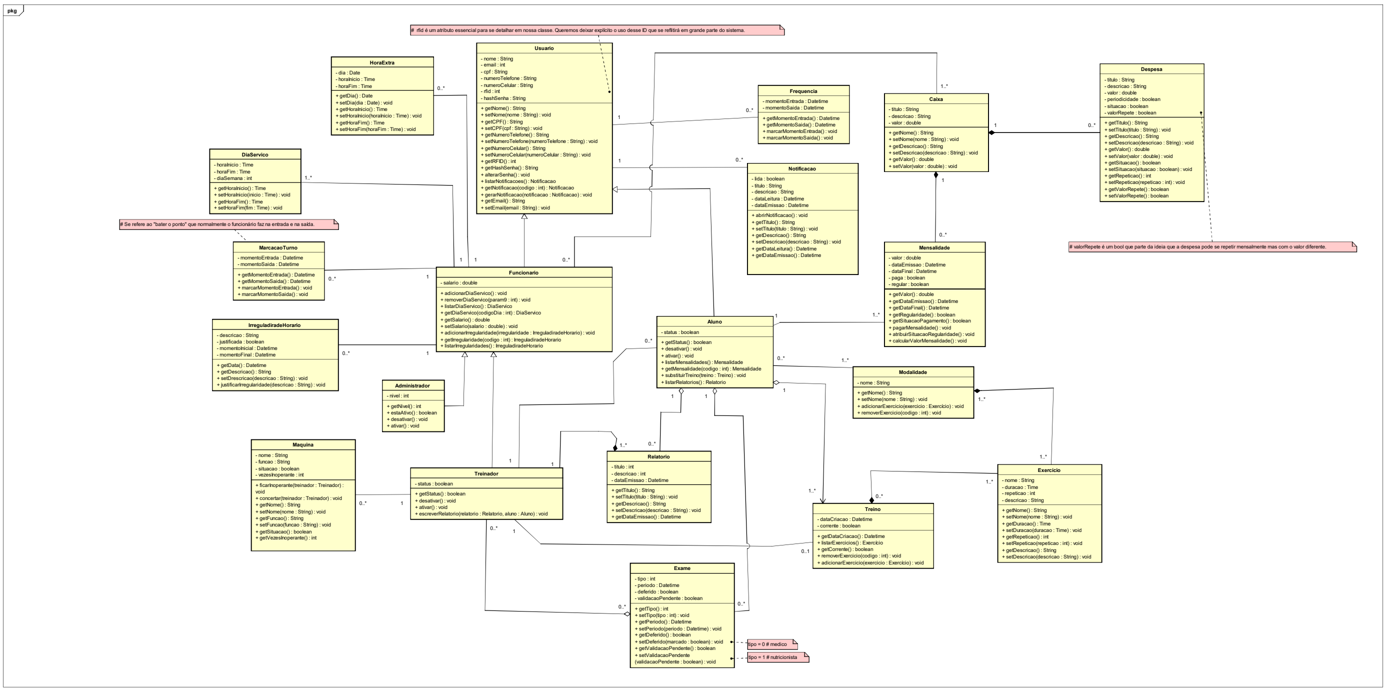
1. O sistema recebe uma resposta válida sobre a operação informando as credenciais incorretas;
2. O sistema retorna para o aluno o status HTTP 400 (Bad request);
3. A aplicação do usuário informa o problema dizendo que ocorreu um problema nas informações do meio de pagamento;
4. O caso de uso é abortado.

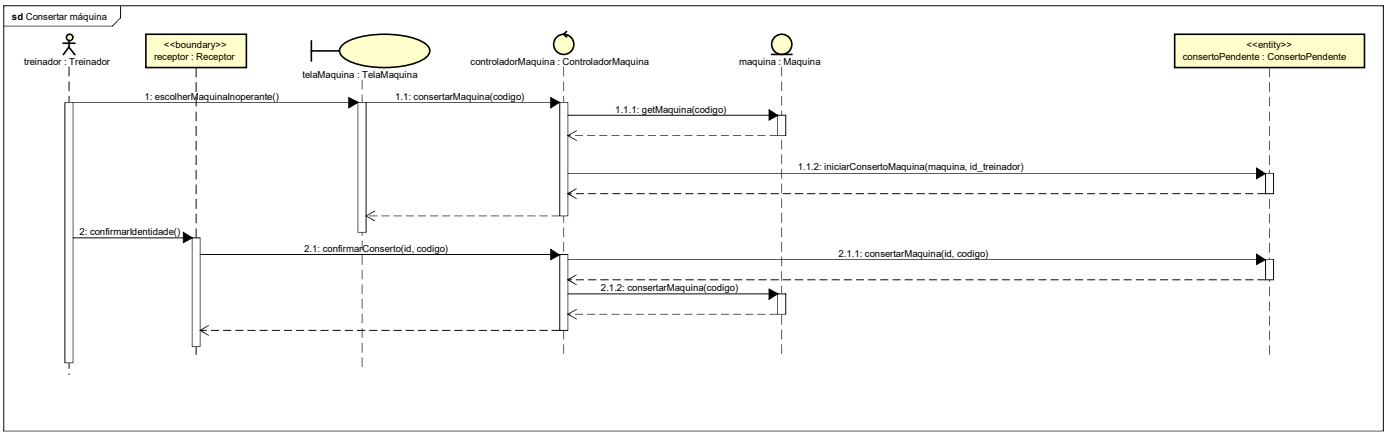
Pré-condições

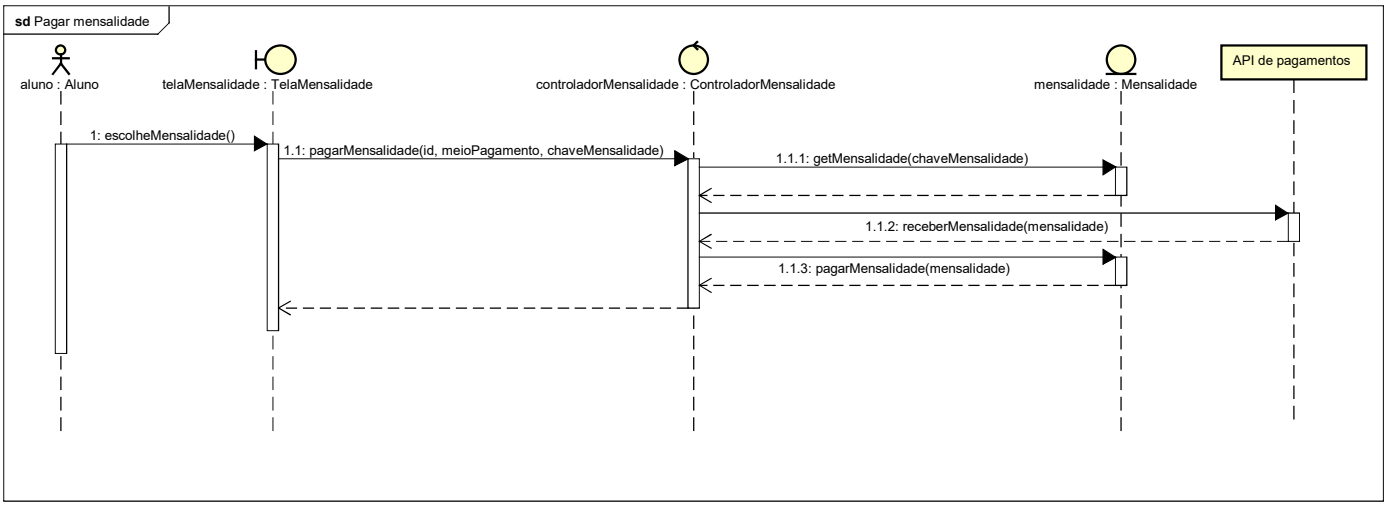
1. O aluno deve estar autenticado no sistema e informar um meio de pagamento válido;
2. A API de Pagamentos deve estar em operação.

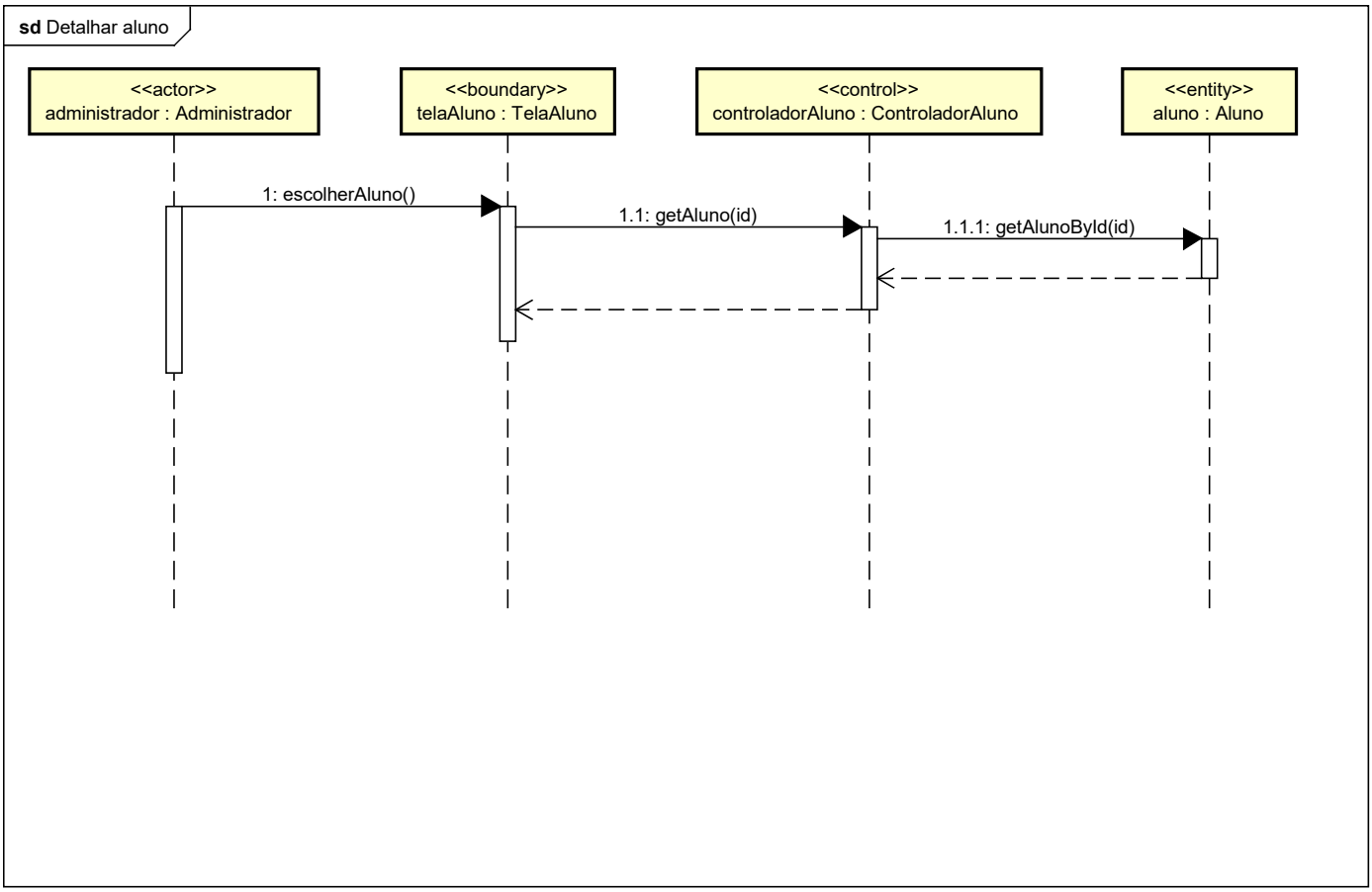
Pós-condição

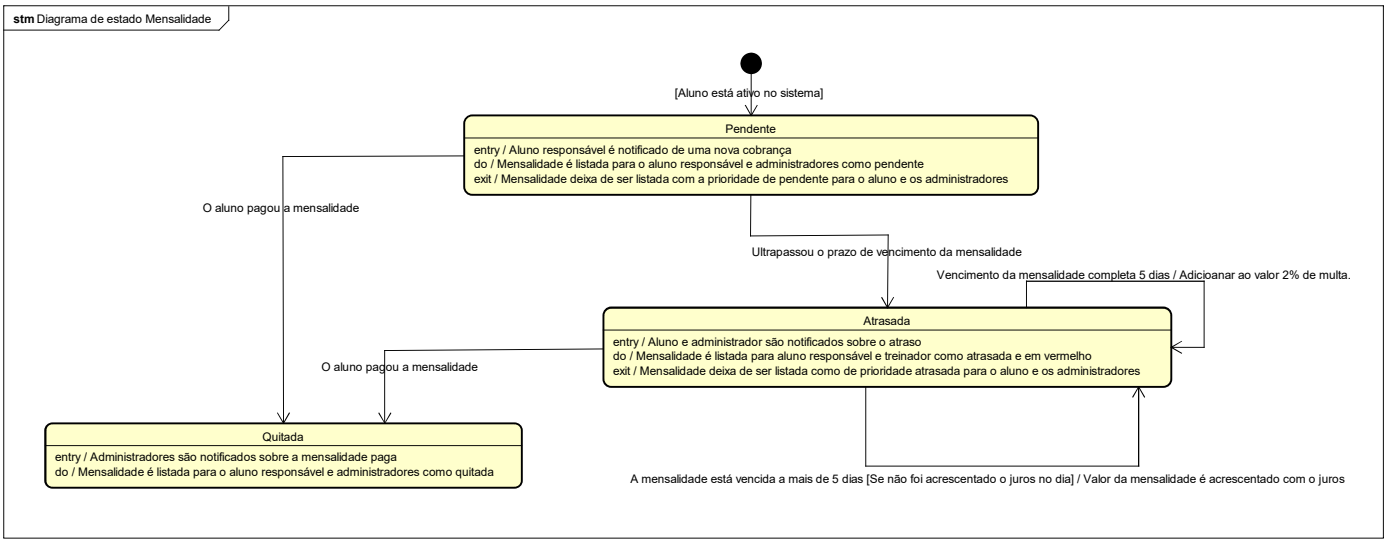
1. O aluno terá uma mensalidade quitada (True) e regularizada (True).

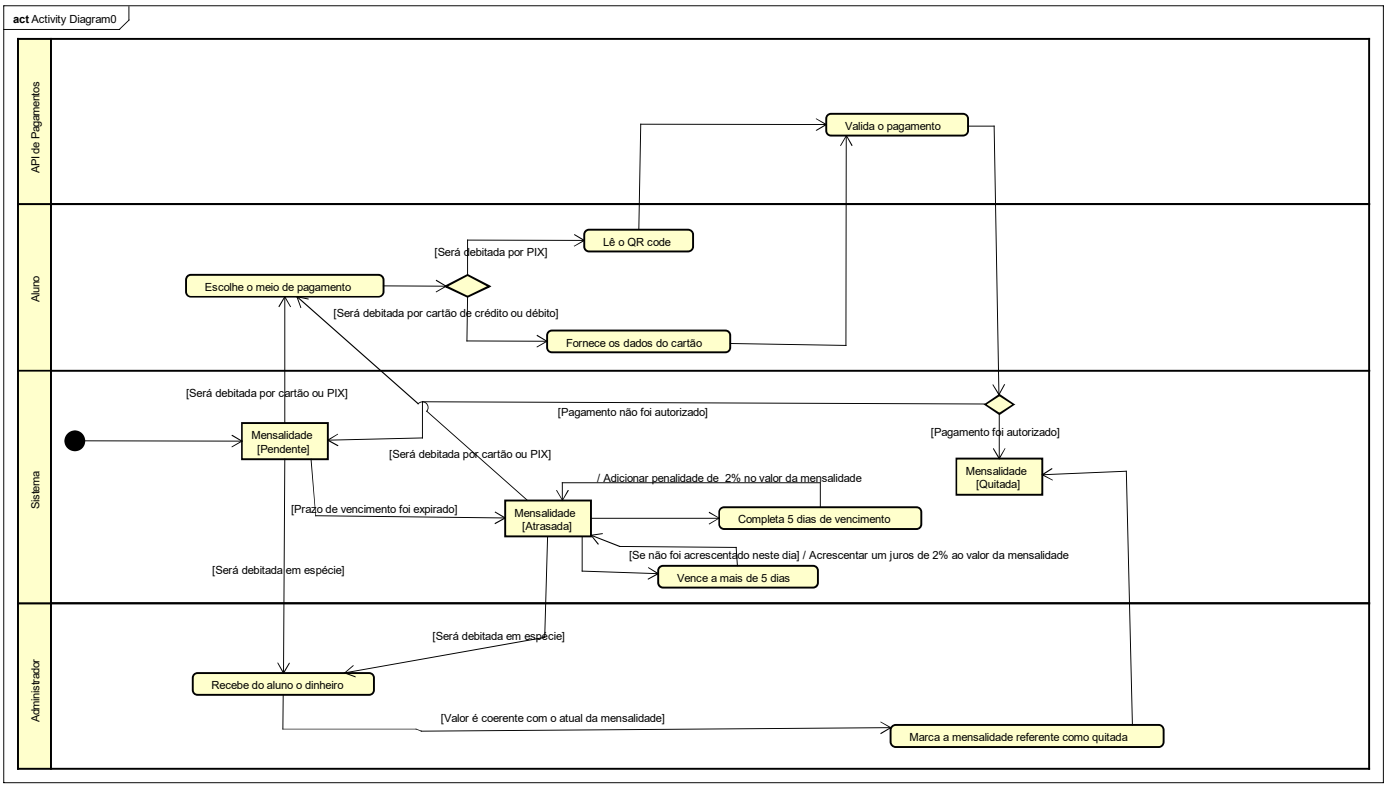

Usuario
HoraExtra
DiaServico
MarcacaoTurno
IrregularidadeHorario
Funcionario
Administrador
Maquina
Treinador
Relatorio
Aluno
Exame
Frequencia
Notificacao
Caixa
Mensalidade
Modalidade
Exercicio
Treino
Despesa

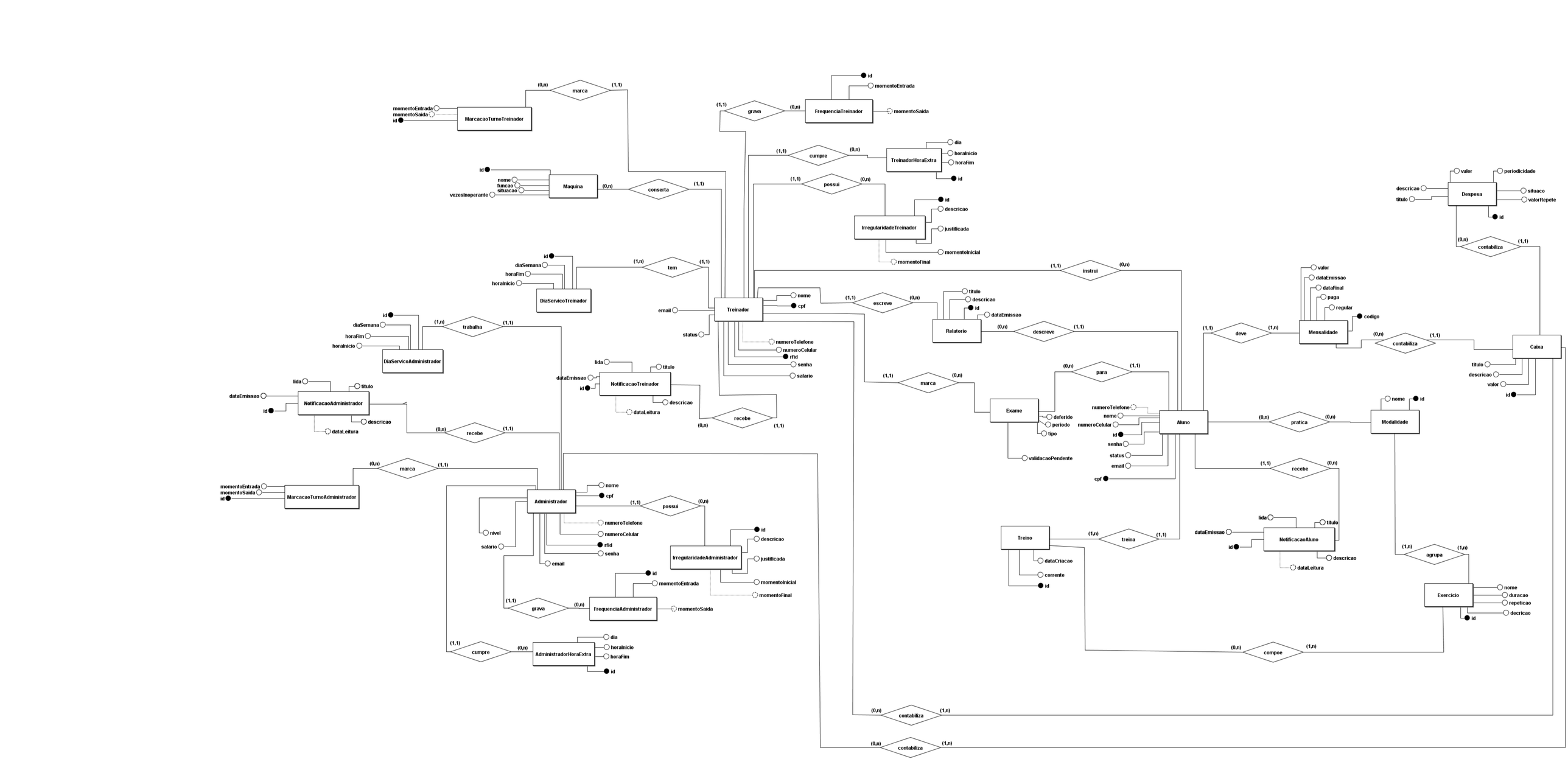

sd Consertar máquina
treinador : Treinador
<<boundary>>
receptor : Receptor
telaMaquina : TelaMaquina
controladorMaquina : ControladorMaquina
maquina : Maquina
<<entity>>
consertoPendente : ConsertoPendente
1: escolherMaquinaInoperante()
1.1: consertarMaquina(codigo)
1.1.1: getMaquina(codigo)
1.1.2: iniciarConsertoMaquina(maquina, id_treinador)
2: confirmarIdentidade()
2.1: confirmarConserto(id, codigo)
2.1.1: consertarMaquina(id, codigo)
2.1.2: consertarMaquina(codigo)

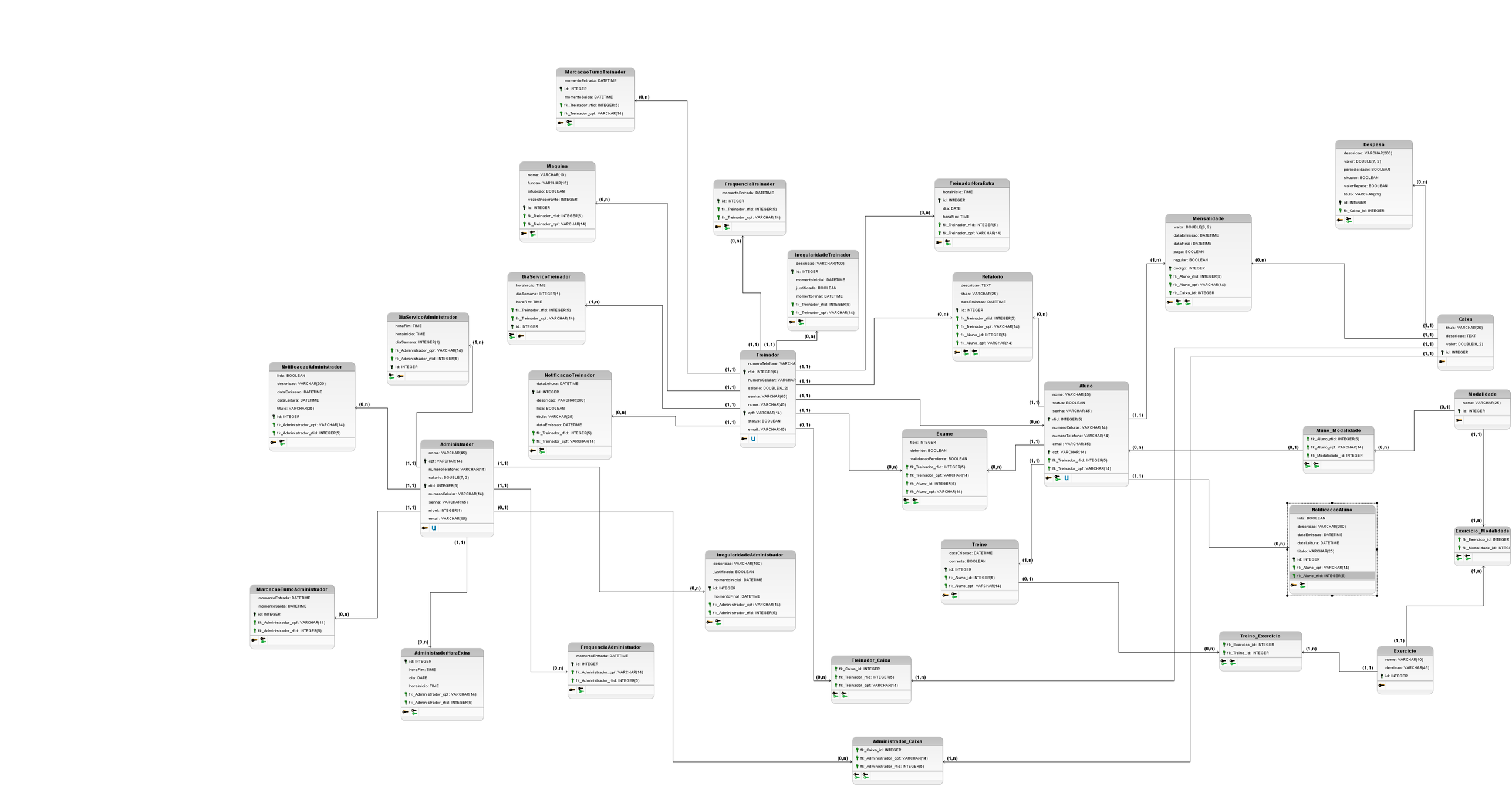

sd Pagar mensalidade
aluno : Aluno
telaMensalidade : TelaMensalidade
controladorMensalidade : ControladorMensalidade
mensalidade : Mensalidade
API de pagamentos
1: escolheMensalidade()
1.1: pagarMensalidade(id, meioPagamento, chaveMensalidade)
1.1.1: getMensalidade(chaveMensalidade)
1.1.2: receberMensalidade(mensalidade)
1.1.3: pagarMensalidade(mensalidade)


sd Detalhar aluno
<<actor>>
administrador : Administrador
<<boundary>>
telaAluno : TelaAluno
<<control>>
controladorAluno : ControladorAluno
<<entity>>
aluno : Aluno
1: escolherAluno()
1.1: getAluno(id)
1.1.1: getAlunoById(id)


stm Diagrama de estado Mensalidade
[Aluno está ativo no sistema]
Pendente
entry / Aluno responsável é notificado de uma nova cobrança
do / Mensalidade é listada para o aluno responsável e administradores como pendente
exit / Mensalidade deixa de ser listada com a prioridade de pendente para o aluno e os administradores
O aluno pagou a mensalidade
Ultrapassou o prazo de vencimento da mensalidade
Vencimento da mensalidade completa 5 dias / Adicioanar ao valor 2% de multa.
Atrasada
entry / Aluno e administrador são notificados sobre o atraso
do / Mensalidade é listada para aluno responsável e treinador como atrasada e em vermelho
exit / Mensalidade deixa de ser listada como de prioridade atrasada para o aluno e os administradores
O aluno pagou a mensalidade
Quitada
entry / Administradores são notificados sobre a mensalidade paga
do / Mensalidade é listada para o aluno responsável e administradores como quitada
A mensalidade está vencida a mais de 5 dias [Se não foi acrescentado o juros no dia] / Valor da mensalidade é acrescentado com o juros


act Activity Diagram0
API de Pagamentos
Aluno
Sistema
Administrador
Valida o pagamento
Lê o QR code
[Será debitada por PIX]
Escolhe o meio de pagamento
[Será debitada por cartão de crédito ou débito]
Fornece os dados do cartão
[Será debitada por cartão ou PIX]
[Pagamento não foi autorizado]
[Pagamento foi autorizado]
Mensalidade [Pendente]
[Será debitada por cartão ou PIX]
Mensalidade [Quitada]
/ Adicionar penalidade de 2% no valor da mensalidade
[Prazo de vencimento foi expirado]
Mensalidade [Atrasada]
Completa 5 dias de vencimento
[Se não foi acrescentado neste dia] / Acrescentar um juros de 2% ao valor da mensalidade
[Será debitada em espécie]
Vence a mais de 5 dias
[Será debitada em espécie]
Recebe do aluno o dinheiro
[Valor é coerente com o atual da mensalidade]
Marca a mensalidade referente como quitada


MarcacaoTurnoTreinador
momentoEntrada
momentoSaida
id
marca
(0,n)
(1,1)
Maquina
nome
funcao
situacao
vezesInoperante
id
conserta
(0,n)
(1,1)
DiaServicoTreinador
id
diaSemana
horaFim
horaInicio
tem
(1,n)
(1,1)
DiaServicoAdministrador
id
diaSemana
horaFim
horaInicio
trabalha
(1,n)
(1,1)
NotificacaoAdministrador
lida
titulo
dataEmissao
id
descricao
dataLeitura
recebe
(0,n)
(1,1)
NotificacaoTreinador
lida
titulo
dataEmissao
id
descricao
dataLeitura
recebe
(0,n)
(1,1)
Treinador
nome
cpf
email
status
numeroTelefone
numeroCelular
rfid
senha
salario
grava
(1,1)
(0,n)
FrequenciaTreinador
id
momentoEntrada
momentoSaida
cumpre
(1,1)
(0,n)
TreinadorHoraExtra
dia
horaInicio
horaFim
id
possui
(1,1)
(0,n)
IrregularidadeTreinador
id
descricao
justificada
momentoInicial
momentoFinal
escreve
(1,1)
(0,n)
Relatorio
titulo
descricao
id
dataEmissao
descreve
(0,n)
(1,1)
instrui
(1,1)
(0,n)
marca
(1,1)
(0,n)
Exame
deferido
periodo
tipo
validacaoPendente
para
(0,n)
(1,1)
Despesa
valor
periodicidade
descricao
titulo
situaco
valorRepete
id
contabiliza
(0,n)
(1,1)
Mensalidade
valor
dataEmissao
dataFinal
paga
regular
codigo
deve
(1,1)
(1,n)
contabiliza
(0,n)
(1,1)
Caixa
titulo
descricao
valor
id
Aluno
numeroTelefone
nome
numeroCelular
id
senha
status
email
cpf
pratica
(0,n)
(0,n)
Modalidade
nome
id
recebe
(1,1)
(0,n)
Treino
dataCriacao
corrente
id
treina
(1,n)
(1,1)
NotificacaoAluno
lida
titulo
dataEmissao
id
descricao
dataLeitura
agrupa
(1,n)
(1,n)
Exercicio
nome
duracao
repeticao
decricao
id
compoe
(0,n)
(1,n)
contabiliza
(0,n)
(1,n)
contabiliza
(0,n)
(1,n)
marca
(0,n)
(1,1)
MarcacaoTurnoAdministrador
momentoEntrada
momentoSaida
id
Administrador
nome
cpf
nivel
salario
numeroTelefone
numeroCelular
rfid
senha
email
possui
(1,1)
(0,n)
IrregularidadeAdministrador
id
descricao
justificada
momentoInicial
momentoFinal
grava
(1,1)
(0,n)
FrequenciaAdministrador
id
momentoEntrada
momentoSaida
cumpre
(1,1)
(0,n)
AdministradorHoraExtra
dia
horaInicio
horaFim
id


MarcacaoTumoTreinador
momentoEntrada: DATETIME
id: INTEGER
momentoSaida: DATETIME
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
Maquina
nome: VARCHAR(10)
funcao: VARCHAR(15)
situacao: BOOLEAN
vezesInoperante: INTEGER
id: INTEGER
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
FrequenciaTreinador
momentoEntrada: DATETIME
id: INTEGER
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
TreinadorHoraExtra
horaInicio: TIME
id: INTEGER
dia: DATE
horaFim: TIME
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
Despesa
descricao: VARCHAR(200)
valor: DOUBLE(7, 2)
periodicidade: BOOLEAN
situaco: BOOLEAN
valorRepete: BOOLEAN
titulo: VARCHAR(25)
id: INTEGER
fk_Caixa_id: INTEGER
IrregularidadeTreinador
descricao: VARCHAR(100)
id: INTEGER
momentoInicial: DATETIME
justificada: BOOLEAN
momentoFinal: DATETIME
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
Mensalidade
valor: DOUBLE(6, 2)
dataEmissao: DATETIME
dataFinal: DATETIME
paga: BOOLEAN
regular: BOOLEAN
codigo: INTEGER
fk_Aluno_rfid: INTEGER(5)
fk_Aluno_cpf: VARCHAR(14)
fk_Caixa_id: INTEGER
DiaServicoTreinador
horaInicio: TIME
diaSemana: INTEGER(1)
horaFim: TIME
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
id: INTEGER
Relatorio
descricao: TEXT
titulo: VARCHAR(25)
dataEmissao: DATETIME
id: INTEGER
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
fk_Aluno_id: INTEGER(5)
fk_Aluno_cpf: VARCHAR(14)
Caixa
titulo: VARCHAR(25)
descricao: TEXT
valor: DOUBLE(8, 2)
id: INTEGER
DiaServicoAdministrador
horaFim: TIME
horaInicio: TIME
diaSemana: INTEGER(1)
fk_Administrador_cpf: VARCHAR(14)
fk_Administrador_rfid: INTEGER(5)
id: INTEGER
NotificacaoAdministrador
lida: BOOLEAN
descricao: VARCHAR(200)
dataEmissao: DATETIME
dataLeitura: DATETIME
titulo: VARCHAR(25)
id: INTEGER
fk_Administrador_cpf: VARCHAR(14)
fk_Administrador_rfid: INTEGER(5)
NotificacaoTreinador
dataLeitura: DATETIME
id: INTEGER
descricao: VARCHAR(200)
lida: BOOLEAN
titulo: VARCHAR(25)
dataEmissao: DATETIME
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
Treinador
numeroTelefone: VARCHA
rfid: INTEGER(5)
numeroCelular: VARCHAR
salario: DOUBLE(6, 2)
senha: VARCHAR(65)
nome: VARCHAR(45)
cpf: VARCHAR(14)
status: BOOLEAN
email: VARCHAR(45)
U
Aluno
nome: VARCHAR(45)
status: BOOLEAN
senha: VARCHAR(45)
rfid: INTEGER(5)
numeroCelular: VARCHAR(14)
numeroTelefone: VARCHAR(14)
email: VARCHAR(45)
cpf: VARCHAR(14)
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
U
Modalidade
nome: VARCHAR(25)
id: INTEGER
Exame
tipo: INTEGER
deferido: BOOLEAN
validacaoPendente: BOOLEAN
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
fk_Aluno_id: INTEGER(5)
fk_Aluno_cpf: VARCHAR(14)
Aluno_Modalidade
fk_Aluno_rfid: INTEGER(5)
fk_Aluno_cpf: VARCHAR(14)
fk_Modalidade_id: INTEGER
Administrador
nome: VARCHAR(45)
cpf: VARCHAR(14)
numeroTelefone: VARCHAR(14)
salario: DOUBLE(7, 2)
rfid: INTEGER(5)
numeroCelular: VARCHAR(14)
senha: VARCHAR(65)
nivel: INTEGER(1)
email: VARCHAR(45)
U
NotificacaoAluno
lida: BOOLEAN
descricao: VARCHAR(200)
dataEmissao: DATETIME
dataLeitura: DATETIME
titulo: VARCHAR(25)
id: INTEGER
fk_Aluno_cpf: VARCHAR(14)
fk_Aluno_rfid: INTEGER(5)
Exercicio_Modalidade
fk_Exercicio_id: INTEGER
fk_Modalidade_id: INTEGE
Treino
dataCriacao: DATETIME
corrente: BOOLEAN
id: INTEGER
fk_Aluno_id: INTEGER(5)
fk_Aluno_cpf: VARCHAR(14)
IrregularidadeAdministrador
descricao: VARCHAR(100)
justificada: BOOLEAN
momentoInicial: DATETIME
id: INTEGER
momentoFinal: DATETIME
fk_Administrador_cpf: VARCHAR(14)
fk_Administrador_rfid: INTEGER(5)
MarcacaoTumoAdministrador
momentoEntrada: DATETIME
momentoSaida: DATETIME
id: INTEGER
fk_Administrador_cpf: VARCHAR(14)
fk_Administrador_rfid: INTEGER(5)
AdministradorHoraExtra
id: INTEGER
horaFim: TIME
dia: DATE
horaInicio: TIME
fk_Administrador_cpf: VARCHAR(14)
fk_Administrador_rfid: INTEGER(5)
FrequenciaAdministrador
momentoEntrada: DATETIME
id: INTEGER
fk_Administrador_cpf: VARCHAR(14)
fk_Administrador_rfid: INTEGER(5)
Treinador_Caixa
fk_Caixa_id: INTEGER
fk_Treinador_rfid: INTEGER(5)
fk_Treinador_cpf: VARCHAR(14)
Treino_Exercicio
fk_Exercicio_id: INTEGER
fk_Treino_id: INTEGER
Exercicio
nome: VARCHAR(10)
decricao: VARCHAR(45)
id: INTEGER
Administrador_Caixa
fk_Caixa_id: INTEGER
fk_Administrador_cpf: VARCHAR(14)
fk_Administrador_rfid: INTEGER(5)

Script: Modelo Físico do HaLoc

```
/*Database*/
create database if not exists haloc;
use haloc;

/* model: */
CREATE TABLE IF NOT EXISTS Administrador (
    nome VARCHAR(45),
    cpf VARCHAR(14),
    numeroTelefone VARCHAR(14),
    salario DOUBLE(7, 2),
    rfid INTEGER(5),
    numeroCelular VARCHAR(14),
    senha VARCHAR(65),
    nivel INTEGER(1),
    email VARCHAR(45),
    PRIMARY KEY (cpf, rfid),
    UNIQUE (cpf, rfid)
);

CREATE TABLE IF NOT EXISTS DiaServicoTreinador (
    horaInicio TIME,
    diaSemana INTEGER(1),
    horaFim TIME,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14),
    id INTEGER PRIMARY KEY
);

CREATE TABLE IF NOT EXISTS Treinador (
    numeroTelefone VARCHAR(8),
    rfid INTEGER(5),
    numeroCelular VARCHAR(14),
    salario DOUBLE(6, 2),
    senha VARCHAR(65),
    nome VARCHAR(45),
    cpf VARCHAR(14),
    status BOOLEAN,
    email VARCHAR(45),
    PRIMARY KEY (rfid, cpf),
    UNIQUE (rfid, cpf)
);

CREATE TABLE IF NOT EXISTS IrregularidadeAdministrador (
    descricao VARCHAR(100),
```

```
    justificada BOOLEAN,
    momentoInicial DATETIME,
    id INTEGER PRIMARY KEY,
    momentoFinal DATETIME,
    fk_Administrador_cpf VARCHAR(14),
    fk_Administrador_rfid INTEGER(5)
);

CREATE TABLE IF NOT EXISTS IrregularidadeTreinador (
    descricao VARCHAR(100),
    id INTEGER PRIMARY KEY,
    momentoInicial DATETIME,
    justificada BOOLEAN,
    momentoFinal DATETIME,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14)
);

CREATE TABLE IF NOT EXISTS FrequenciaAdministrador (
    momentoEntrada DATETIME,
    id INTEGER PRIMARY KEY,
    fk_Administrador_cpf VARCHAR(14),
    fk_Administrador_rfid INTEGER(5)
);

CREATE TABLE IF NOT EXISTS TreinadorHoraExtra (
    horaInicio TIME,
    id INTEGER PRIMARY KEY,
    dia DATE,
    horaFim TIME,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14)
);

CREATE TABLE IF NOT EXISTS AdministradorHoraExtra (
    id INTEGER PRIMARY KEY,
    horaFim TIME,
    dia DATE,
    horaInicio TIME,
    fk_Administrador_cpf VARCHAR(14),
    fk_Administrador_rfid INTEGER(5)
);

CREATE TABLE IF NOT EXISTS FrequenciaTreinador (
    momentoEntrada DATETIME,
    id INTEGER PRIMARY KEY,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14)
```

```
);

CREATE TABLE IF NOT EXISTS NotificacaoAdministrador (
    lida BOOLEAN,
    descricao VARCHAR(200),
    dataEmissao DATETIME,
    dataLeitura DATETIME,
    titulo VARCHAR(25),
    id INTEGER PRIMARY KEY,
    fk_Administrador_cpf VARCHAR(14),
    fk_Administrador_rfid INTEGER(5)
);

CREATE TABLE IF NOT EXISTS NotificacaoTreinador (
    dataLeitura DATETIME,
    id INTEGER PRIMARY KEY,
    descricao VARCHAR(200),
    lida BOOLEAN,
    titulo VARCHAR(25),
    dataEmissao DATETIME,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14)
);

CREATE TABLE IF NOT EXISTS Aluno (
    nome VARCHAR(45),
    status BOOLEAN,
    senha VARCHAR(45),
    rfid INTEGER(5),
    numeroCelular VARCHAR(14),
    numeroTelefone VARCHAR(14),
    email VARCHAR(45),
    cpf VARCHAR(14),
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14),
    PRIMARY KEY (rfid, cpf),
    UNIQUE (rfid, cpf)
);

CREATE TABLE IF NOT EXISTS Maquina (
    nome VARCHAR(10),
    funcao VARCHAR(15),
    situacao BOOLEAN,
    vezesInoperante INTEGER,
    id INTEGER PRIMARY KEY,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14)
);
```

```
CREATE TABLE IF NOT EXISTS Relatorio (
    descricao TEXT,
    titulo VARCHAR(25),
    dataEmissao DATETIME,
    id INTEGER PRIMARY KEY,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14),
    fk_Aluno_id INTEGER(5),
    fk_Aluno_cpf VARCHAR(14)
);

CREATE TABLE IF NOT EXISTS Mensalidade (
    valor DOUBLE(6, 2),
    dataEmissao DATETIME,
    dataFinal DATETIME,
    paga BOOLEAN,
    regular BOOLEAN,
    codigo INTEGER PRIMARY KEY,
    fk_Aluno_rfid INTEGER(5),
    fk_Aluno_cpf VARCHAR(14),
    fk_Caixa_id INTEGER
);

CREATE TABLE IF NOT EXISTS Treino (
    dataCriacao DATETIME,
    corrente BOOLEAN,
    id INTEGER PRIMARY KEY,
    fk_Aluno_id INTEGER(5),
    fk_Aluno_cpf VARCHAR(14)
);

CREATE TABLE IF NOT EXISTS Modalidade (
    nome VARCHAR(25),
    id INTEGER PRIMARY KEY
);

CREATE TABLE IF NOT EXISTS Exercicio (
    nome VARCHAR(10),
    decricao VARCHAR(45),
    id INTEGER PRIMARY KEY
);

CREATE TABLE IF NOT EXISTS Caixa (
    titulo VARCHAR(25),
    descricao TEXT,
    valor DOUBLE(8, 2),
    id INTEGER PRIMARY KEY
```

```
);

CREATE TABLE IF NOT EXISTS Despesa (
    descricao VARCHAR(200),
    valor DOUBLE(7, 2),
    periodicidade BOOLEAN,
    situaco BOOLEAN,
    valorRepete BOOLEAN,
    titulo VARCHAR(25),
    id INTEGER PRIMARY KEY,
    fk_Caixa_id INTEGER
);

CREATE TABLE IF NOT EXISTS DiaServicoAdministrador (
    horaFim TIME,
    horaInicio TIME,
    diaSemana INTEGER(1),
    fk_Administrador_cpf VARCHAR(14),
    fk_Administrador_rfid INTEGER(5),
    id INTEGER PRIMARY KEY
);

CREATE TABLE IF NOT EXISTS MarcacaoTurnoAdministrador (
    momentoEntrada DATETIME,
    momentoSaida DATETIME,
    id INTEGER PRIMARY KEY,
    fk_Administrador_cpf VARCHAR(14),
    fk_Administrador_rfid INTEGER(5)
);

CREATE TABLE IF NOT EXISTS MarcacaoTurnoTreinador (
    momentoEntrada DATETIME,
    id INTEGER PRIMARY KEY,
    momentoSaida DATETIME,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14)
);

CREATE TABLE IF NOT EXISTS Exame (
    tipo INTEGER,
    deferido BOOLEAN,
    validacaoPendente BOOLEAN,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14),
    fk_Aluno_id INTEGER(5),
    fk_Aluno_cpf VARCHAR(14)
);
```

```
CREATE TABLE IF NOT EXISTS Aluno_Modalidade (
    fk_Aluno_rfid INTEGER(5),
    fk_Aluno_cpf VARCHAR(14),
    fk_Modalidade_id INTEGER
);

CREATE TABLE IF NOT EXISTS Exercicio_Modalidade (
    fk_Exercico_id INTEGER,
    fk_Modalidade_id INTEGER
);

CREATE TABLE IF NOT EXISTS Treino_Exercicio (
    fk_Exercico_id INTEGER,
    fk_Treino_id INTEGER
);

CREATE TABLE IF NOT EXISTS Treinador_Caixa (
    fk_Caixa_id INTEGER,
    fk_Treinador_rfid INTEGER(5),
    fk_Treinador_cpf VARCHAR(14)
);

CREATE TABLE IF NOT EXISTS Administrador_Caixa (
    fk_Caixa_id INTEGER,
    fk_Administrador_cpf VARCHAR(14),
    fk_Administrador_rfid INTEGER(5)
);

CREATE TABLE IF NOT EXISTS NotificacaoAluno (
    lida BOOLEAN,
    descricao VARCHAR(200),
    dataEmissao DATETIME,
    dataLeitura DATETIME,
    titulo VARCHAR(25),
    id INTEGER PRIMARY KEY,
    fk_Aluno_cpf VARCHAR(14),
    fk_Aluno_rfid INTEGER(5)
);

/*ADICIONANDO FOREIGN KEY NAS TABELAS CRIADAS*/
ALTER TABLE DiaServicoTreinador ADD CONSTRAINT FK_DiaServicoTreinador_1
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE RESTRICT;

ALTER TABLE IrregularidadeAdministrador ADD CONSTRAINT
FK_IrregularidadeAdministrador_2
    FOREIGN KEY (fk_Administrador_cpf, fk_Administrador_rfid)
```

```
    REFERENCES Administrador (cpf, rfid)
    ON DELETE CASCADE;

ALTER TABLE IrregularidadeTreinador ADD CONSTRAINT FK_IrregularidadeTreinador_2
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE FrequenciaAdministrador ADD CONSTRAINT
FK_FrequenciaAdministrador_2
    FOREIGN KEY (fk_Administrador_cpf, fk_Administrador_rfid)
    REFERENCES Administrador (cpf, rfid)
    ON DELETE CASCADE;

ALTER TABLE TreinadorHoraExtra ADD CONSTRAINT FK_TreinadorHoraExtra_2
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE AdministradorHoraExtra ADD CONSTRAINT FK_AdministradorHoraExtra_2
    FOREIGN KEY (fk_Administrador_cpf, fk_Administrador_rfid)
    REFERENCES Administrador (cpf, rfid)
    ON DELETE CASCADE;

ALTER TABLE FrequenciaTreinador ADD CONSTRAINT FK_FrequenciaTreinador_2
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE NotificacaoAdministrador ADD CONSTRAINT
FK_NotificacaoAdministrador_2
    FOREIGN KEY (fk_Administrador_cpf, fk_Administrador_rfid)
    REFERENCES Administrador (cpf, rfid)
    ON DELETE CASCADE;

ALTER TABLE NotificacaoTreinador ADD CONSTRAINT FK_NotificacaoTreinador_2
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE NotificacaoAluno ADD CONSTRAINT FK_NotificacaoAluno_3
    FOREIGN KEY (fk_Aluno_rfid, fk_Aluno_cpf)
    REFERENCES Aluno (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE Aluno ADD CONSTRAINT FK_Aluno_2
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
```

```
    ON DELETE CASCADE;

ALTER TABLE Maquina ADD CONSTRAINT FK_Maquina_2
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE Relatorio ADD CONSTRAINT FK_Relatorio_2
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE Relatorio ADD CONSTRAINT FK_Relatorio_3
    FOREIGN KEY (fk_Aluno_id, fk_Aluno_cpf)
    REFERENCES Aluno (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE Mensalidade ADD CONSTRAINT FK_Mensalidade_2
    FOREIGN KEY (fk_Aluno_rfid, fk_Aluno_cpf)
    REFERENCES Aluno (rfid, cpf)
    ON DELETE RESTRICT;

ALTER TABLE Mensalidade ADD CONSTRAINT FK_Mensalidade_3
    FOREIGN KEY (fk_Caixa_id)
    REFERENCES Caixa (id)
    ON DELETE CASCADE;

ALTER TABLE Treino ADD CONSTRAINT FK_Treino_2
    FOREIGN KEY (fk_Aluno_id, fk_Aluno_cpf)
    REFERENCES Aluno (rfid, cpf)
    ON DELETE RESTRICT;

ALTER TABLE Despesa ADD CONSTRAINT FK_Despesa_2
    FOREIGN KEY (fk_Caixa_id)
    REFERENCES Caixa (id)
    ON DELETE CASCADE;

ALTER TABLE DiaServicoAdministrador ADD CONSTRAINT
FK_DiaServicoAdministrador_1
    FOREIGN KEY (fk_Administrador_cpf, fk_Administrador_rfid)
    REFERENCES Administrador (cpf, rfid)
    ON DELETE RESTRICT;

ALTER TABLE MarcacaoTurnoAdministrador ADD CONSTRAINT
FK_MarcacaoTurnoAdministrador_2
    FOREIGN KEY (fk_Administrador_cpf, fk_Administrador_rfid)
    REFERENCES Administrador (cpf, rfid)
    ON DELETE CASCADE;
```

```
ALTER TABLE MarcacaoTurnoTreinador ADD CONSTRAINT
FK_MarcacaoTurnoTreinador_2
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE Exame ADD CONSTRAINT FK_Exame_1
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE Exame ADD CONSTRAINT FK_Exame_2
    FOREIGN KEY (fk_Aluno_id, fk_Aluno_cpf)
    REFERENCES Aluno (rfid, cpf)
    ON DELETE CASCADE;

ALTER TABLE Aluno_Modalidade ADD CONSTRAINT FK_Aluno_Modalidade_1
    FOREIGN KEY (fk_Aluno_rfid, fk_Aluno_cpf)
    REFERENCES Aluno (rfid, cpf)
    ON DELETE SET NULL;

ALTER TABLE Aluno_Modalidade ADD CONSTRAINT FK_Aluno_Modalidade_2
    FOREIGN KEY (fk_Modalidade_id)
    REFERENCES Modalidade (id)
    ON DELETE SET NULL;

ALTER TABLE Exercicio_Modalidade ADD CONSTRAINT FK_Exercicio_Modalidade_1
    FOREIGN KEY (fk_Exercico_id)
    REFERENCES Exercicio (id)
    ON DELETE RESTRICT;

ALTER TABLE Exercicio_Modalidade ADD CONSTRAINT FK_Exercicio_Modalidade_2
    FOREIGN KEY (fk_Modalidade_id)
    REFERENCES Modalidade (id)
    ON DELETE RESTRICT;

ALTER TABLE Treino_Exercicio ADD CONSTRAINT FK_Treino_Exercicio_1
    FOREIGN KEY (fk_Exercico_id)
    REFERENCES Exercicio (id)
    ON DELETE RESTRICT;

ALTER TABLE Treino_Exercicio ADD CONSTRAINT FK_Treino_Exercicio_2
    FOREIGN KEY (fk_Treino_id)
    REFERENCES Treino (id)
    ON DELETE SET NULL;

ALTER TABLE Treinador_Caixa ADD CONSTRAINT FK_Treinador_Caixa_1
```

```
    FOREIGN KEY (fk_Caixa_id)
    REFERENCES Caixa (id)
    ON DELETE RESTRICT;

ALTER TABLE Treinador_Caixa ADD CONSTRAINT FK_Treinador_Caixa_2
    FOREIGN KEY (fk_Treinador_rfid, fk_Treinador_cpf)
    REFERENCES Treinador (rfid, cpf)
    ON DELETE SET NULL;

ALTER TABLE Administrador_Caixa ADD CONSTRAINT FK_Administrador_Caixa_1
    FOREIGN KEY (fk_Caixa_id)
    REFERENCES Caixa (id)
    ON DELETE RESTRICT;

ALTER TABLE Administrador_Caixa ADD CONSTRAINT FK_Administrador_Caixa_2
    FOREIGN KEY (fk_Administrador_cpf, fk_Administrador_rfid)
    REFERENCES Administrador (cpf, rfid)
    ON DELETE SET NULL;
```