O projeto que desenvolvemos é a modelagem de um sistema de controle de acesso e gestão de capacidade para academias, A modelagem segue o paradigma NoSQL Abstract Model (NoAM), buscando uma estrutura flexível e de alto desempenho.

Justificativa:

No contexto atual, gerenciar o fluxo de pessoas e garantir o limite de ocupação de um ambiente é crucial, especialmente para o setor *fitness*. Nosso projeto visa solucionar a ineficiência de métodos manuais ou sistemas legados de check-in, que falham em fornecer dados em tempo real. A adoção do NoAM permite que o sistema lide com a alta volumetria de dados de *Check-in* e *logs* de forma eficiente, além de oferecer a flexibilidade necessária para que a entidade Academia possa evoluir com novos atributos sem grandes reestruturações de *schema*. Essencialmente, estamos propondo uma base de dados que prioriza a velocidade de leitura e a rastreabilidade histórica.

Objetivos:

1. Modelar um banco de dados NoSQL robusto com, no mínimo, cinco entidades interconectadas, respeitando as diretrizes de cardinalidade e tipo de ligação.
2. Criar um registro de `Checkin` que seja totalmente rastreável, vinculando-o ao `Cliente` e à `Academia` com precisão de data e hora.
3. Estruturar a entidade `Academia` para manter o **`current_occupancy`** (ocupação atual) atualizado em tempo real, fornecendo um dado vital para a gestão.
4. Implementar a segregação de responsabilidades administrativas através da entidade `Usuário` e seu campo **`role`**, garantindo controle de acesso adequado.

Descrição das Entidades e Seus Campos

A modelagem é composta por cinco entidades centrais, cada uma com um papel específico no ecossistema do sistema de gestão. A notação de **Obrigatório** indica campos que são ***NOT NULL*** (não podem estar vazios) no momento da criação ou atualização do registro.

Entidade: Academia

Representa uma unidade física da academia. É a entidade central de controle de capacidade e gestão.

Campo	Tipo	Obrigatoriedade	Descrição
id	`ObjectId`	Obrigatório	Chave primária única da unidade.
nome	`String`	Obrigatório	Nome de identificação comercial da unidade.
telefone	`String`	Opcional	Telefone de contato da unidade.
owner_id	`ObjectId`	Obrigatório	Referência ao `Usuário` principal (gerente ou dono) da unidade.
current_occupancy	`Integer`	Obrigatório	Contagem em tempo real de clientes atualmente dentro da unidade.
endereco_id	`ObjectId`	Obrigatório	Chave estrangeira que aponta para o endereço físico da academia.

Entidade: Usuário

Representa os perfis administrativos com acesso ao sistema de gestão (funcionários, gerentes, etc.).

Campo	Tipo	Obrigatoriedade	Descrição
id	`ObjectId`	Obrigatório	Chave primária única do usuário administrativo.
username	`String`	Obrigatório	Nome de login do usuário, deve ser único no sistema.
senha	`String`	Obrigatório	Credencial de segurança para acesso, geralmente armazenada como hash.
role	`String`	Obrigatório	Define o nível de permissão (ex.: 'Admin', 'Gerente', 'Recepcionista').

Entidade: Cliente

Representa a pessoa física que é membro ou frequentador da academia.

Campo	Tipo	Obrigatoriedade	Descrição
id	`ObjectId`	Obrigatório	Chave primária única do cliente.
nome	`String`	Obrigatório	Nome completo para identificação.

email	`String`	Obrigatório	Principal meio de contato e, em alguns casos, login.
telefone	`String`	Opcional	Número secundário de contato.
endereco	`ObjectId`	Opcional	Referência ao seu endereço residencial.

Entidade: Endereço

Armazena dados geográficos, permitindo o reuso e a normalização dos dados de localização.

Campo	Tipo	Obrigatoriedade	Descrição
id	`ObjectId`	Obrigatório	Chave primária única do registro de endereço.
cep	`String`	Obrigatório	Código postal para localização precisa.
rua	`String`	Obrigatório	Nome da rua ou logradouro.

Entidade: Checkin

Representa o evento histórico de entrada de um cliente em uma unidade da academia.

Campo	Tipo	Obrigatoriedade	Descrição
id	`ObjectId`	Obrigatório	Chave primária única do registro do evento.
cliente_id	`ObjectId`	Obrigatório	Chave estrangeira que aponta para o cliente que está fazendo o check-in.
gym_id	`ObjectId`	Obrigatório	Chave estrangeira que aponta para a academia acessada.
checkin_time	`Date`	Obrigatório	O timestamp exato do momento da entrada.

is_active	`Boolean`	Opcional	Indicador que pode ser usado para marcar a saída (*checkout*), embora o evento principal seja a entrada.

Justificativa das Relações e Cardinalidades

As relações entre as entidades foram escolhidas para refletir o ciclo de vida e a dependência lógica dos dados, conforme a arquitetura NoSQL.

Relação 1: Academia ↔ Endereço

- **Tipo Escolhido: Composição**
- **Justificativa da Ligação:** Utilizamos a **Composição** (losango preenchido) pois o endereço é considerado uma parte *integrante* e *exclusiva* da unidade `Academia`. Uma Academia não pode existir semanticamente sem um local físico. Se a Academia for excluída do sistema, seu endereço correspondente também deve ser logicamente excluído, pois não fará mais sentido de forma isolada.
- **Cardinalidade (1:1):** Uma Academia está localizada em **exatamente um** endereço, e um registro de `Endereço` neste contexto pertence **somente a uma** Academia (não é compartilhado entre unidades).

Relação 2: Academia ↔ Usuário

- **Tipo Escolhido: Agregação**
- **Justificativa da Ligação:** Optamos pela **Agregação** (losango vazio) pois o `Usuário` (gerente, por exemplo) tem um ciclo de vida independente da `Academia`. O Usuário pode ser transferido ou gerenciar múltiplas unidades. A Academia *agrega* o Usuário em sua gestão, mas se a unidade fechar, o Usuário continua existindo no sistema.
- **Cardinalidade (1:0..*):** Uma Academia é gerenciada por **um ou mais** usuários (relação de `owner_id`), e um Usuário pode estar vinculado a **zero ou mais** Academias.

Relação 3: Cliente ↔ Checkin

- **Tipo Escolhido: Composição**
- **Justificativa da Ligação:** O registro de **`Checkin` é um evento dependente** que só tem significado no contexto de um `Cliente`. É uma **Composição** porque, se o registro do Cliente for permanentemente excluído (o que é raro, mas possível),

todos os seus eventos de Check-in passados também devem ser removidos para manter a integridade dos dados históricos.

- **Cardinalidade (1:0..*):** Um Cliente tem a capacidade de realizar **zero ou vários** Check-ins ao longo de sua permanência, e cada registro de Check-in pertence a **exatamente um** Cliente.

Relação 4: Cliente ↔ Endereço

- **Tipo Escolhido: Associação**
- **Justificativa da Ligação:** Esta é uma **Associação** simples porque o endereço é apenas um dado referencial do cliente. O cliente pode ter ou não um endereço cadastrado e, ao se mudar, a referência é atualizada sem afetar o ciclo de vida do cliente ou do registro de endereço (que pode ser reusado).
- **Cardinalidade (1:1):** Um Cliente possui **zero ou um** endereço residencial no sistema, e um registro de Endereço só pode estar associado a **um** Cliente (para evitar que um único registro de endereço seja mapeado para várias pessoas).

Relação 5: Academia ↔ Checkin

- **Tipo Escolhido: Associação**
- **Justificativa da Ligação:** Embora o `Checkin` ocorra *na* `Academia`, ele já está sob forte dependência do `Cliente` (Composição). Portanto, esta é uma **Associação** simples que apenas indica qual unidade foi o local do evento.
- **Cardinalidade (1:0..*):** Uma Academia registra **zero ou vários** eventos de Check-in ao longo do tempo, e um Check-in sempre ocorre em **exatamente uma** unidade de Academia.