Universidade do Minho
Licenciatura em Engenharia Informática

Unidade Curricular de Bases de Dados

Ano Lectivo de 2022/2023

Motivation Gym

João Manuel Machado Lopes (A100594)

Tiago Nuno de Magalhães Teixeira (A100665)

Miguel Jacinto Dias Carvalho (A84518)

João Carlos Viana Pereira Marques, 84684

B

Resumo

Este projeto foi desenvolvido pelo grupo 50, no âmbito da unidade curricular de Bases de Dados. No mundo atual, uma gestão eficiente de dados é essencial para o sucesso de qualquer empreendimento. No caso de um ginásio, a gestão adequada das informações dos membros, funcionários, aulas e equipamentos é fundamental para garantir um serviço de qualidade e proporcionar uma experiência positiva aos utilizadores.

Uma base de dados em SQL para um ginásio tem como objetivo principal criar uma estrutura organizada para armazenar e manipular esses dados de forma eficiente. Através do uso da linguagem SQL, foi possível definir as tabelas, relacionamentos e consultas necessárias para atender às necessidades específicas do ginásio.

A implementação de queries SQL possibilitou a obtenção de informações relevantes de forma precisa e rápida. Foi possível, por exemplo, listar todos os clientes inscritos no ginásio ou encontrar o instrutor responsável por uma determinada aula.

Estes são apenas alguns exemplos da solução estruturada pelo nosso grupo.

Índice

1. Introdução

1.1. Contextualização

O senhor Júnior, um homem de 32 anos, nunca esteve satisfeito com a sua aparência física, contudo, nunca teve a possibilidade de treinar num ginásio convencional, uma vez que o seu trabalho e vida familiar não lhe permitiam conciliar. O único desporto que este praticava eram as suas caminhadas de 30 minutos no trajeto de casa para o trabalho e vice-versa, não sendo estas suficientes para atingir os objetivos que ambicionava.

Todavia, com o surgimento da Covid-19 e de todas as restrições a si associadas, Júnior alterou o seu regime profissional para teletrabalho e, por isso, acabou por abandonar as suas caminhadas. Cada vez mais sedentário, evidenciaram-se ainda mais os efeitos colaterais da falta de exercício na sua vida. Sentia constante falta de ar, desmotivação contínua, baixa autoestima e cansaço extremo. Isto permaneceria assim até meados de 2021, momento este que foi ao médico de família numa consulta de rotina e este o confrontou com uma realidade assustadora. O senhor Júnior havia ganho diabetes tipo 2 e a sua concentração de colesterol que se devia situar abaixo dos 200 mg/dl, encontrava-se a 300 mg/dl. Diante destes fatos e das consequências que estes podiam vir a carregar, como o aumento significativo do risco de doenças cardiovasculares e até mesmo, em casos extremos, a necessidade de amputar membros, decidiu tomar uma atitude.

Aconselhou-se de especialistas na área do fitness e comprou um pequeno conjunto de halteres, uma barra de musculação, um banco de inclinação variável, uns tapetes, alguns conjuntos de roupa desportiva e transformou a sua garagem no seu pequeno ginásio moldado a seu gosto, necessidades e recomendações de profissionais. Sem qualquer conhecimento na área, optou por contratar um *personal trainer* durante as primeiras semanas, que o acompanharia presencialmente uma vez por semana, à sexta-feira, e o convidaria para sessões de grupo através do *ZOOM* nos restantes dias, retirando, desta forma, o melhor aproveitamento possível do seu ginásio recém-criado e transmitindo-lhe a confiança de que estava no caminho certo.

Entre os seus treinos e o trabalho, o senhor Júnior motivado pelos resultados que vinha obtendo e pelo gosto que havia ganho pela área, começou a reservar algum tempo do seu dia para aprofundar o seu conhecimento na mesma, lendo sobre anatomia funcional, psicologia do desporto, fisiologia do exercício e muitos outros temas relacionados à atividade física.

Eventualmente o confinamento terminou e, com ele, também os problemas de saúde do senhor Júnior haviam melhorado, o seu colesterol situava-se agora nos 170 mg/dl e os diabetes encontravam-se sob controle. Contudo, não se deixou acomodar e continuou o seu regime de treino, tendo apenas adaptado este aos dias em que não podia realizar o seu trabalho em casa, passando, nestes dias, a treinar de madrugada, não reduzindo assim ao horário de trabalho nem ao tempo passado com a família. Com o conhecimento que havia adquirindo e juntamente com a sua experiência, decidiu que estaria na altura de pôr estes em prática, em prol de transformar a vida de pessoas que estariam na situação que ele outrora se encontrou.

Decidiu então convidar dois amigos que sempre o apoiaram na sua jornada, mas que nunca embarcaram nela. Comprou mais equipamentos e investiu num melhor sistema de isolamento e de som, visando melhorar o ambiente no seu ginásio caseiro sem perturbar a paz dos seus entes queridos. Durante algumas semanas foram apenas os três, mas com os resultados a tornar-se cada vez mais evidentes, os amigos não resistiram a espalhar a notícia de que o senhor Júnior tinha um ginásio em casa, que andava a treiná-los e que a suas vidas havia e estavam a mudar para melhor. Nesse ponto, o senhor Júnior decidiu fazer uma introspeção e mudar o rumo da sua vida.

Foi então a 10 de janeiro de 2022, que decidiu demitir-se do seu emprego como gestor de marketing e dedicar-se a tempo inteiro à sua paixão, ser um *personal trainer*. Combinando toda a reputação e popularidade que adquiriu através dos seus amigos e a experiência profissional que possuía na área do marketing, não tinha dúvidas que ia ser bem-sucedido. E verdade seja dita, não estava errado, pois primeiro inscreveram-se os amigos, depois os amigos dos amigos e quando deu por si, já estava a acompanhar clientes de cidades vizinhas. Fazia todo o tipo de acompanhamento, dava aulas privadas, aulas de grupo, acompanhamento online e ainda compartilhava dicas e o seu estilo de vida nas redes sociais. Proporcional a este crescimento e às necessidades do seu negócio, foi preenchendo o seu ginásio com novos equipamentos, de forma a oferecer um serviço de excelência e assegurar o bem-estar de todos os clientes.

Eventualmente, com dezenas de clientes inscritos, a garagem e a sua agenda atingiram o seu limite. De forma a quebrar esta barreira, investiu na construção de um anexo no seu quintal com lotação máxima de 20 pessoas e contratou os seus primeiros funcionários, dois experientes *personal trainers* que havia conhecido através das suas campanhas de marketing digital e a sua mulher como secretária, controlando os pagamentos e o agendamento dos clientes.

Isto manter-se-ia durante 3 meses, até atingir um novo limite. Nesse momento, não querendo abrandar o crescimento exponencial da empresa, os senhores Júnior em conjunto com a sua mulher decidiram que estaria na hora de expandir o negócio para além do terreno da sua casa.

Alugou um armazém com 450 metros quadrados, contratou mais 3 *personal trainers*, mais 1 secretária, 2 responsáveis pela limpeza do local, 1 técnico de reparações e manutenção dos equipamentos, comprou ainda mais equipamentos de musculação e montou um novo ginásio, a que chamaria “Motivation Gym”. Desta forma, não só conseguiu aumentar a capacidade do local, como também proporcionar uma melhor experiência aos utilizadores, disponibilizando balneários femininos e masculinos, máquinas de bebidas e comida e mais casas de banho, algo que era bastante limitado em sua casa, pois não havia sido construída visando servir tantas pessoas.

Hoje, o único problema que enfrenta, com o afluxo de novos clientes e funcionários, é a dificuldade no controle e gerenciamento dos dados dos clientes e funcionários. Em média, cada secretária perde entre 3 e 5 minutos a dar entrada a um cliente, pois o processo de verificação da inscrição e do pagamento atualizado do cliente é demorado, fazendo com que se criem filas na entrada do ginásio. Surgem de igual modo atrasos associados ao agendamento de aulas privadas e de grupo, pois é trabalhoso confirmar o *personal trainer* com disponibilidade. Entre outros problemas.

O senhor Júnior precisa, portanto, de uma melhor forma de gerir os dados necessários ao normal funcionamento do seu negócio, abandonando assim os registos tradicionais em papel.

1.2. Fundamentação

As bases de dados são de extrema importância para todos os negócios que possuam um elevado volume de operadores e operandos. Uma correta implementação de uma base de dados permite uma visão precisa sobre todos os setores da empresa, permitindo assim transmitir aos proprietários informações relevantes sobre qualquer setor desejado, o que é útil, por exemplo, para avaliar falhas de informação e incoerências de valores face aos esperados.

No caso do ginásio “Motivation Gym”, presenças dos membros, assim como as suas mensalidades, os seus planos de atividades, avaliações nutricionais e informações sobre os funcionários eram guardadas em formato de papel, uma vez que a dimensão do negócio ainda não era significativamente grande.

No entanto, com o aumento do número de clientes e funcionários, o senhor Júnior começou a receber queixas acerca da má gestão das aulas de grupo dadas, assim como incoerências na faturação. Além disso, o mesmo não conseguia viabilizar corretamente os diferentes planos de treino, cujo preço varia em função do horário escolhido pelos usuários, já que o controle das entradas e saídas não era minucioso o suficiente.

1.3. Objetivos

Baseado na sua experiência com o negócio e clientes, o senhor Júnior decidiu que pretendia alcançar os seguintes objetivos com a implementação da futura base de dados no seu ginásio:

- Organizar o seu modelo de negócio.
- Gerir cuidadosamente a recolha de informações sobre as entradas e saídas de cada utilizador no ginásio, de modo a permitir a distinção nos preços das mensalidades praticadas, baseados no horário em que o utilizador pode frequentar o ginásio com base na mensalidade paga.
- Monitorizar as faturas emitidas por cada cliente e funcionário do ginásio, de forma a administrar minuciosamente os pagamentos e evitar quaisquer possíveis erros ou problemas de cobrança, bem como manter um registo detalhado das transações financeiras, relatórios financeiros e auditorias, aumentando a satisfação do cliente e a confiança na gestão financeira do negócio.
- Facilitar o processo de agendamento das aulas de grupos e privadas, a verificação da sua lotação e o respetivo treinador que vai monitorá-la.
- Conhecer melhor os clientes e perceber as suas preferências relativas às aulas de grupo, bem como os seus horários nos quais eles treinam.
- Determinar quais as aulas e horários mais frequentados, permitindo, assim, mobilizar mais treinadores para esses, fornecendo um melhor acompanhamento, uma maior atratividade e um melhor serviço.

1.5. Viabilidade

Se o senhor Júnior substituir o seu atual método de registo em papel por um formato diferente de forma a eliminar possíveis erros de faturação, controlo de acessos e gestão de lotações de aulas, o mesmo conseguirá:

- Reduzir a 100% entradas não autorizadas e fora de horário, permitindo apenas a entrada a clientes dentro do período estabelecido no contrato aquando da inscrição.
- Aumentar em 5% os lucros mensais, relacionado ao melhor controlo das entradas e saídas, pois obriga os clientes que extrapolam os seus horários, a alterarem o seu contrato para um com um horário mais expandido, caso pretendam, de facto, continuar a ir no mesmo horário.
- Desta forma, reduz o número de críticas relativas a falta de disponibilidade e assistência profissional e aumenta em 30% a participação de clientes em aulas de grupo.
- Controlar a lotação, fazendo flutuar o valor das mensalidades consoante a percentagem da mesma.

1.6. Recursos

Humanos: Treinadores com formação para aulas de grupo e/ou *personal-training*, clientes e funcionários da empresa de desenvolvimento.

Materiais:

Hardware (1 servidor, 1 estabelecimento).

Software (*SGBD* e Aplicações de Vendas e Aprovisionamento).

1.7. Equipa de Trabalho

Pessoal interno: Senhor Júnior, treinadores e rececionista, para ajudar no levantamento de requisitos, validação de serviços e funcionamento do ginásio.

Pessoal externo: Engenheiros de *software* responsáveis por levantar requisitos, modelar, desenvolver e implementar a base de dados.

Outros: Inquéritos de opinião realizados para averiguação da satisfação dos clientes face aos serviços prestados.

1.8. Plano de execução

Com o intuito de definir e planear a forma como o processo de desenvolvimento da base de dados irá ser realizado, o senhor Júnior, com a ajuda da sua recentemente contratada secretária, a senhora Vera, fez o plano de trabalho, agrupando temporalmente as diferentes tarefas a serem realizadas tal como quem as irá realizar, com o utensílio de um diagrama de *Gantt*.

2. Levantamento e Análise de requisitos

2.1. Método de levantamento e de análise de requisitos adotado

Com o intuito de tornar a futura base de dados o mais prática, funcional e completa possível, de forma a suprimir da melhor forma todas as necessidades do negócio do Senhor Júnior, usamos as seguintes estratégias para recolher os requisitos:

- Analise dos registos que o Senhor Júnior usava até então, usados para armazenar todos os dados relativos ao ginásio e aos clientes e funcionários que engloba, de forma a identificar e registar os dados que seriam necessários guardar na base de dados.
- 2 reuniões com o Senhor Júnior e a sua mulher, discutindo o funcionamento do ginásio, as limitações do método atual e que dados e operações consideravam que ao serem guardados e implementadas facilitariam o seu trabalho e melhorariam o negócio.
- Distribuição de inquéritos aos funcionários, com o objetivo de identificar as dificuldades que o modelo antigo apresentava e o que desejavam ver no modelo novo.
- Realização de inquéritos de opinião aos clientes, de modo a descobrir as falhas do sistema antigo e de conhecer as expectativas dos mesmos face ao modelo futuro.
- Observação direta do ginásio em funcionamento, visando identificar, hierarquizar e identificar os processos que nele decorrem, para assegurar que todas as necessidades são atendidas. Desta forma, confirmamos os dados relevantes a serem guardados na base de dados, obtidos através dos outros métodos, e acrescentamos, se necessário, mais alguns, no caso de, por exemplo, o Senhor Júnior os ter esquecido de mencionar.
- Observação de outros estabelecimentos semelhantes, selecionando ginásios com boa reputação, bem estabelecidos na área e se possível, com sistemas de bases de dados já implementados. Deste modo, aumentamos a nossa confiança nos requisitos já recolhidos, reduzindo eventuais esquecimentos e corrigindo qualquer ambiguidade, ou mesmo, melhorando algum requisito.

2.2. Organização dos requisitos levantados

2.2.1. Requisitos de descrição

- O ginásio é caracterizado por um id único, um nome, uma morada e uma data de abertura;
- Cada ginásio tem vários funcionários, que podem ter diversas funções;
- Existem apenas 5 funções possíveis, sendo estas: treinadores de sala, treinadores privados, treinadores de aulas de grupo, rececionistas e diretor.
- Um funcionário pode ter mais que uma função, mas não são possíveis todas as combinações de funções;
- Cada funcionário deve ter um ID único;
- Cada funcionário tem a si associadas as informações do seu nome, sexo (0 -> feminino; 1 -> Masculino), data de nascimento, morada, número de telemóvel, IBAN, NIF, e-mail e data de contratação;
- Cada funcionário tem ainda associado a si um ginásio onde trabalha, um salário base e um horário de trabalho.
- As moradas, são definidas pela rua, código postal, cidade e distrito;
- Cada ginásio tem vários clientes;
- Os clientes têm um número de identificação único, de forma a identificá-los univocamente;
- Em relação a cada cliente é necessário guardar informações do seu nome, sexo (0 -> feminino; 1 -> Masculino), data de nascimento, morada, número de telemóvel, IBAN, NIF, email e data de contratação;
- É necessário guardar a data de inscrição de cada cliente;
- Em relação ao cliente é ainda guardada, se necessário, alguma informação relevante sobre condições médicas;
- Cada cliente tem a si associado um plano de treino;
- Em relação aos clientes, é sempre registada a data e hora que entram e saem do ginásio e em que ginásio isso ocorreu;
- Cada cliente pode escolher ter 2 refills de água de 500 ml por treino, se pagar mais 1,25€ por mês. Caso contrário, apenas pode encher a garrafa de água na casa de banho;
- Cada cliente que tenha um plano de treino pode usufruir de todo o espaço sem custos adicionais, à exceção de aulas de grupo e aulas personalizadas;
- Os clientes se pretenderem ter a possibilidade de tomar banho, têm de pagar mais 4€ por mês;
- Existem 3 planos de treino e são caracterizados por um nome, pelo seu preço e pelo horário que têm associado;
- Os horários associados aos planos de treino são iguais todos os dias;
- Os planos de treino disponíveis são, o low-cost (7:00 - 13:00), o clássico (13:00-19:00) e o livre (7:00 - 23:00);
- Cada plano de treino inclui apenas a possibilidade de treinar com os professores de sala desse mesmo horário, sendo as aulas de grupo e personal training pagas à parte;
- Cada professor de sala tem um horário semanal fixo, sendo igual todos os dias da semana;
- Os horários possíveis são 3, sendo o das 7:00 às 13:00 designado por 1, o das 13:00 às 18:00 por 2 e o das 18:00 às 23:00 por 3.
- Os professores de sala para além de fazerem sala, podem ainda fazer treino especializado e/ou aulas de grupo, mas somente se tiverem essas especializações;

- Cada aula privada tem a si associada um funcionário com essa função, um cliente, um horário em que começa, um horário em que termina e o seu custo que varia com a duração da mesma;
- As aulas privadas decorrem sempre na sala de musculação e têm como modalidade "Personal Training";
- Cada aula de grupo tem a si associada a sua modalidade, um espaço, um horário de começo, um horário de término, um grupo de clientes e um funcionário com especialidade de professor de grupo;
- Um espaço é caracterizado por um número de 1 a 5, correspondendo a uma das 5 salas reservadas a aulas de grupo que o ginásio possuí;
- O espaço número 0 corresponde à sala de musculação.
- No final de cada mês é automaticamente emitida para cada cliente uma fatura que contém o valor total a ser pago por cada cliente, considerado o seu plano de treino, os extras que o ginásio oferece e as opcionais aulas de grupo e personal training;
- Os métodos de pagamento aceites são dinheiro vivo, multibanco e MBWay, representados, respetivamento, por 0, 1 e 2;
- Visto que são registadas as entradas e saídas dos clientes, estas são representadas, respetivamente, por um 0 e um 1;
- A fatura contém ainda um id único, a data de emissão da mesma, a data em que o cliente pagou e o método de pagamento;
- Em relação aos extras que cada cliente tem direito, caso este não opte por nenhum extra, é lhe associado o número 0, se optar somente por água o número 1, somente por banho o número 2 e se optar por ambos o número 3;
- Os treinadores recebem 25% do valor das aulas de grupo ou privadas, sendo este valor adicionado ao seu salário que irão receber no final do respetivo mês.
- O diretor do ginásio é o Senhor Júnior.

2.2.2. Requisitos de exploração

- Em cada instante, saber quantos clientes se encontram no ginásio.
- Em cada instante, saber quais clientes se encontram no ginásio.
- Aceder ao histórico das horas de entrada e saída de um determinado cliente.
- Obter a lista de clientes que estão a usufruir de um determinado plano de treino.
- Fazer um relatório do número de clientes que facultam aulas de grupo.
- Perceber, a cada momento, quantos utilizadores estão inscritos ou já estiveram inscritos em treinos privados.
- Aceder ao histórico de aulas privadas e aulas de grupo de um cliente.
- Em cada momento, deve ser possível verificar as aulas de grupo e privadas que um cliente agendou.
- Fazer um relatório do número e percentagem de utentes que usufruem dos extras que o ginásio dá a possibilidade de aderir, como o refill de água e o banho.
- Saber quantos novos clientes o ginásio angariou em cada mês.
- Saber quantos e quais funcionários de uma determinada especialidade se encontram atualmente ativos, de momento.
- Aceder ao histórico de aulas privadas e/ou de grupo que um funcionário com essas especialidades realizou.
- Em cada momento, consultar as aulas de grupo e/ou privadas que um funcionário com essas especialidades tem agendadas.
- Conhecer quantas aulas privadas ou de grupo cada treinador com essas especialidades deu.
- Entender qual a especialidade que está a render mais lucro para o ginásio.
- Em cada momento, consultar quais os clientes que se encontram inscritos em cada aula de grupo.

2.2.3. Requisitos de controlo

- Um cliente que não tenha escolhido pagar o valor extra associado ao refill de água, não o pode executar.
- Um cliente que não tenha escolhido pagar o valor extra associado à possibilidade de tomar banho no ginásio, não o pode realizar.
- Os clientes apenas podem entrar no ginásio num horário que seja compatível com o plano que estão a pagar.
- Apenas os clientes que estejam inscritos em aulas de grupo podem comparecer às mesmas.
- Os clientes não podem estar inscritos em duas aulas de grupo que decorram em horários concorrentes.
- Cada cliente apenas pode aceder aos seus dados pessoais, tendo para isso de solicitar o acesso a uma das reacionistas.
- Os clientes podem dar múltiplas entradas e saídas do ginásio no mesmo dia.
- Apenas as rececionistas podem editar os dados dos clientes e emitir faturas.
- Apenas as rececionistas podem inscrever novos clientes.
- Apenas as rececionistas podem inscrever os clientes nas aulas de grupo.

- Somente as rececionistas podem remover a inscrição de um cliente numa aula de grupo, se o cliente desejar.
- As aulas de grupo só podem ser lecionadas por professores com essa especialidade.
- Cada treinador tem apenas acesso ao seu horário de trabalho, não tendo acesso ao dos restantes membros.
- As aulas de personal training apenas são lecionadas a clientes que paguem por esse mesmo serviço.
- Cada treinador tem apenas acesso ao seu horário de trabalho, não tendo acesso ao dos restantes membros.
- Apenas os funcionários que tenham especialidade de personal training podem realizar treinos privados.
- O agendamento e não comparticipação de um cliente na aula privada, implica o pagamento da mesma.
- A adição, edição e remoção de funcionários das diversas especialidades pode apenas ser feita pelo senhor Júnior.
- O único diretor é o senhor Júnior.
- Somente o senhor Júnior pode alterar os termos e condições dos contratos dos clientes, caso estes pretendam alterá-los.
- Apenas o senhor Júnior tem acesso às estatísticas, relatórios e todo o tipo de operações que envolvam relacionar os dados de dois ou mais clientes ou funcionários.

2.3. Análise e validação dos requisitos

Acabada a fase de organização e revisão dos requisitos, a equipa de desenvolvimento reuniu-se com os restantes intervenientes do projeto e reviram, detalhada e minuciosamente, cada um dos requisitos, confirmando o seu conteúdo e assegurando a sua correta categorização, satisfazendo assim todos os elementos da equipa.

Após a correção de algumas pontualidades excecionais, todos os elementos presentes validaram os requisitos, unanimemente.

3. Modelação Conceptual

3.1. Apresentação da abordagem de modelação realizada

Tendo em conta os requisitos elaborados no capítulo anterior, tentamos desenvolver um modelo conceptual. Este modelo consiste no uso de entidades, em que cada uma das entidades será caracterizada pelos seus atributos. É necessário também uma peça que faça a ligação entre as entidades, neste caso serão os relacionamentos.

3.2. Identificação e caracterização das entidades

Entidade	Descrição	Ocorrências
Funcionário	Representação de um funcionário que trabalha para o ginásio do senhor Júnior.	Cada funcionário tem uma ou mais funções, assim como vencimentos e pode ser responsável de treinar 0 ou mais clientes.
Função	Representa as várias funções que um funcionário pode desempenhar.	Uma função tem um ou mais funcionários responsáveis por a desempenhar.
Ginásio	Entidade que guarda toda a informação que representa o ginásio do senhor Júnior.	O ginásio tem um ou mais clientes, assim como funcionários, aos quais paga os seus vencimentos.
Fatura	Representa as a faturas pagas pelos clientes do ginásio.	Cada fatura tem apenas um cliente associado.
Cliente	Representação de quem usa os serviços do ginásio do senhor Júnior.	Cada cliente tem um ou mais planos de treino, assim como uma ou mais faturas e pode ser treinado por um funcionário.
Plano de Treino	Representação do horário e valor a pagar dos planos de treino em que os clientes estão inscritos.	Cada plano de treino tem um e um só cliente.

Tabela 1 - Identificação e caraterização de entidades.

3.3. Identificação e caracterização dos relacionamentos

Entidade	Relacionamento	Entidade	Cardinalidade	Participação
Funcionário	Tem	Vencimento	N:M	T:P
Funcionário	Vencimento	Ginásio	1: N	T:P
Funcionário	Treina	Cliente	N:M	P:P
Ginásio	Tem	Cliente	N:M	P: T
Cliente	Paga	Fatura	1: N	P: T
Cliente	Tem	Plano de Treino	N:1	T: P

Tabela 2 - Identificação e caraterização dos relacionamentos

3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos.

Atributos	Tipo	Descrição	Tamanho e Formato	Dominio	Nulo
DataContratação	Simples	Dia, mês e ano que remete para a contratação do funcionário.	Date	Qualquer	N
Telemóvel	Multivalorado	Número de telemóvel associado ao contacto do funcionário.	INT	Qualquer	N
NIF	Simples	Número de identificação fiscal do funcionário.	INT	Qualquer	N
Id	Chave Primária Simples	Número de identificação único do funcionário.	INT	Qualquer	N
Nome	Simples	Nome completo do funcionário.	VARCHAR(80)	Qualquer	N
Sexo	Simples	Género do funcionário.	BOOLEAN	0 (Feminino); 1 (Masculino)	S
Email	Simples	Endereço eletrónico do email do funcionário.	VARCHAR(150)	Qualquer	N
DataNascimento	Simples	Data de Nascimento.	DATE	Qualquer	N
Morada	Composto	Morada do funcionário do funcionário.			N
Rua	Simples	Nome da rua, número da porta, ... do funcionário.	VARCHAR(20)	Qualquer	N
Cidade	Simples	Nome da cidade onde reside o funcionário.	VARCHAR(100)	Qualquer	N
CódigoPostal	Simples	Código Postal do funcionário.	VARCHAR(30)	Qualquer	N
Distrito	Simples	Nome do distrito onde reside o funcionário.	VARCHAR(30)	Qualquer	N
Ginásio	Simples	Ginásio onde o funcionário trabalha	INT	Qualquer	1

SalarioBase	Simples	Salário base do funcionário	DOUBLE(8,4)	Qualquer	1250.00
HorarioTrabalho	Simples	Horário de trabalho do funcionário	INT	1 (7:00 às 13:00); 2 (13:00 às 18:00); 3 (18:00 às 23:00)	1

Tabela 3 - Identificação e caracterização dos atributos da entidade Funcionário

Atributos	Tipo	Descrição	Tamanho e Formato	Dominio	Nulo
Nome	Chave Primária Simples	Nome da função que o funcionário ocupa	VARCHAR(25)	Personal Trainer, Professor de Sala, Professor de Grupo, Rececionista, Diretor.	N

Tabela 4 - Identificação e caracterização dos atributos da entidade Função

Atributos	Tipo	Descrição	Tamanho e Formato	Dominio	Nulo
Nome	Simples	Nome do ginásio.	VARCHAR(30)	Qualquer	N
Id	Chave Primária Simples	Identificador do ginásio.	INT	Qualquer	N
DataAbertura	Simples	Data de inauguração do ginásio.	DATE	Qualquer	N
Morada	Composto	Morada do ginásio.			
CódigoPostal	Simples	Código Postal do ginásio.	VARCHAR(20)	Qualquer	N
Rua	Simples	Nome da rua, número da porta, ... do ginásio.	VARCHAR(100)	Qualquer	N
Cidade	Simples	Nome da cidade onde se situa.	VARCHAR(30)	Qualquer	N
Distrito	Simples	Nome do distrito onde se situa.	VARCHAR(30)	Qualquer	N

Tabela 5 - Identificação e caracterização dos atributos da entidade Ginásio

Atributos	Tipo	Descrição	Tamanho e Formato	Dominio	Nulo
ValorMensalidade	Simples	Valor fixo da mensalidade relacionado com o plano de treino e os extras (refill de água e/ou banho) escolhidos.	DECIMAL(8,2)	Qualquer	N
DataEmissão	Simples	Data de emissão da fatura.	DATE	Qualquer	N

Id	Chave Primária Simples	Id da fatura.	INT	Qualquer	N
PreçoTotal	Simples	Preço total da fatura, calculado através da soma do valor da mensalidade e das aulas privadas e de grupo que o cliente realizou.	DECIMAL(8,2)	Qualquer	N
Data Pagamento	Simples	Data em que o cliente pagou.	DATE	Qualquer	S

Tabela 6 - Identificação e caracterização dos atributos da entidade Fatura

Atributos	Tipo	Descrição	Tamanho e Formato	Dominio	Nulo
Id	Chave Primária Simples	Identificador do cliente.	DECIMAL(8,2)	Qualquer	N
Nome	Simples	Nome completo do Cliente	DATE	Qualquer	N
Sexo	Simples	Género do cliente.	INT	Qualquer	N
Telemóvel	Multivalorado	Número de telemóvel associado ao contacto do cliente.	DECIMAL(8,2)	Qualquer	N
Email	Simples	Endereço eletrónico do email do cliente.	DATE	Qualquer	S
NIF	Simples	Número de identificação fiscal do cliente.	INT	Qualquer	N
DataNascimento	Simples	Data de nascimento do cliente.	VARCHAR(80)	Qualquer	N
CondiçõesMédicas	Simples	Frase que descreve as condições médicas de um cliente.	BOOLEAN	Qualquer	S
Extras	Simples	Código relacionado aos opcionais reffil de água e banhos.	INT	Qualquer	N
Morada	Composto	Morada do cliente.			
Rua	Simples	Nome da rua, número da porta, ... do cliente.	VARCHAR(20)	Qualquer	N
Cidade	Simples	Nome da cidade onde reside o cliente.	VARCHAR(100)	Qualquer	N
CódigoPostal	Simples	Código postal do cliente.	VARCHAR(30)	Qualquer	N
Distrito	Simples	Nome do distrito onde reside o cliente.	VARCHAR(30)	Qualquer	N
EntradasSaídas	Multivalorado e Composto	Entradas e saídas de cada cliente			

Data	Simples	Dia, mês e ano em que o cliente entrou ou saiu do ginásio.	DATE	Qualquer	N
Horário	Simples	Hora do dia na qual o	TIME	Qualquer	N
EntradaSaída	Simples	Determina se o cliente entrou ou saiu.	BOOLEAN	0 (Entrada); 1 (Saída)	N

Tabela 7 - Identificação e caracterização dos atributos da entidade Cliente

Atributos	Tipo	Descrição	Tamanho e Formato	Dominio	Nulo
Preço	Chave Primária Simples	Preço do plano de treino.	Decimal (8,2)	Qualquer	N
HorárioInício	Simples	Hora do dia a partir da qual o cliente está autorizado a entrar no ginásio.	TIME	Qualquer	N
HorárioFim	Simples	Hora do dia até à qual o cliente está autorizado a estar no ginásio.	TIME	Qualquer	N

Tabela 8 - Identificação e caracterização dos atributos da entidade Plano de Treino

Atributos	Tipo	Descrição	Tamanho e Formato	Dominio	Nulo
HorárioTrabalho	Simples	Horário de trabalho de um funcionário.	INT	0(Turno da manhã, 7:00 - 13:00); 1(Turno de tarde, 13:00-19:00); 2(Turno da noite, 19:00-23:00)	N
Saláriototal	Derivado	Salário Total (com extras) de um funcionário.	DECIMAL(8,4)	Qualquer	N
SalárioBase	Simples	Salário Base de um funcionário.	DECIMAL(8,4)	Qualquer	N
DataPagamento	Simples	Data de pagamento do salário do funcionário.	DATE	Qualquer	N

Tabela 9 - Identificação e caracterização dos atributos do relacionamento Vencimento

Atributos	Tipo	Descrição	Tamanho e Formato	Dominio	Nulo
HorárioInício	Simples	Horário do início do treino.	DATETIME	Qualquer	N
HorárioFim	Simples	Horário do fim do treino.	DATETIME	Qualquer	N
Espaço	Simples	Sala onde foi realizado o treino.	VARCHAR(30)	Qualquer	N
Modalidade	Simples	Modalidade do treino realizado.	VARCHAR(30)	Qualquer	N
Valor	Simples	Valor unitário do treino	DECIMAL(8,4)	Qualquer	N

Tabela 10 - Identificação e caracterização dos atributos do relacionamento Treina

Atributos	Tipo	Descrição	Tamanho e Formato	Dominio	Nulo
Data de Inscrição	Simples	Data de inscrição de um cliente num ginásio.	DATE	Qualquer	N

Tabela 11 - Identificação e caracterização dos atributos do relacionamento Tem

3.5. Apresentação e explicação do diagrama ER produzido

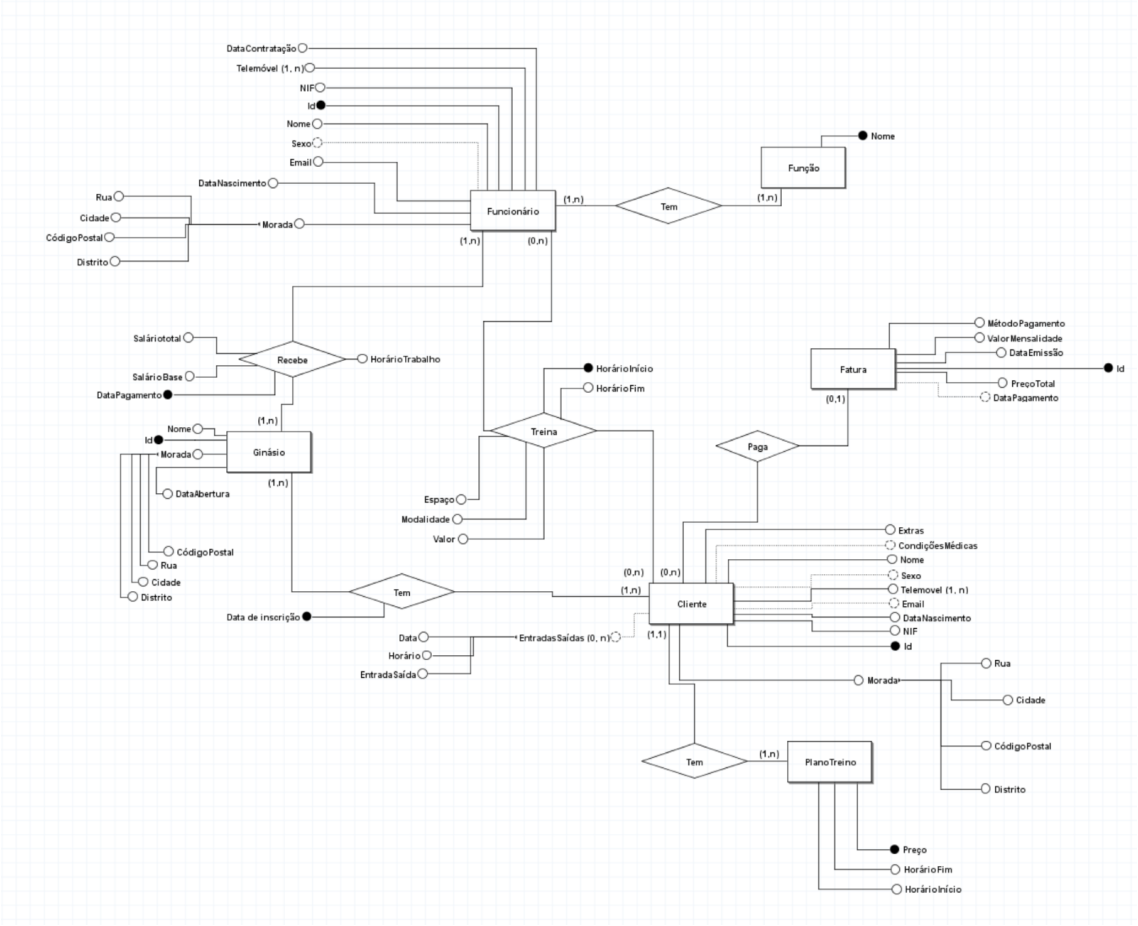


Figura 1 - Modelo Conceptual

Em relação ao mapa conceptual, as entidades e relacionamentos entre as mesmas, foram todos construidos e estabelecidos de forma a responder aos requisitos recolhidos da melhor maneira.

Em relação às entidades, estas têm associadas a si todos os atributos necessários à caracterização das mesmas. Por exemplo, no caso do funcionário, os atributos que este possuí derivaram dos seguintes requisitos:

- Um funcionário pode ter mais que uma função, mas não são possíveis todas as combinações de funções.
- Cada funcionário deve ter um ID único.
- Cada funcionário tem a si associadas as informações do seu nome, sexo (0 -> feminino; 1 -> Masculino), data de nascimento, morada, número de telemóvel, IBAN, NIF, email e data de contratação.
- As moradas, são definidas pela rua, código postal, cidade e distrito.
- Cada funcionário tem ainda associado a si um ginásio onde trabalha, um salário base e um horário de trabalho.

Em relação às demais entidades, os atributos das mesmas foram obtidos da mesma forma.

Partindo para os relacionamentos, temos dois de elevada importância, o denominado de “recebe”, que é estabelecido entre a entidade funcionário e a entidade ginásio e denominado de “treina”, que é estabelecido entre a entidade funcionário e a entidade cliente.

Em relação ao relacionamento “recebe” este pretende responder às necessidades criadas pelos requisitos que referem a existência de faturas (pagamentos aos funcionários). Cada entrada na tabela que será gerada a partir deste relacionamento irá caracerizar uma fatura e terá associada a si os atributos que estão associados ao relacionamento recebe: SalarioBase, HorarioTrabalho, SalarioTotal e dataPagamento. De realçar que a entidade funcionário também possuí os atributos SalarioBase, Ginasio e HorarioTrabalho, isto porque, caso o senhor Júnior decida alterar um destes atributos associados a uma instância de funcionário, as faturas emitidas antes dessa alteração permanecerão intactas, com os dados correspondentes a esse período, assegurando assim uma correcta representação da realidade e a coerência temporal.

Por outro lado, em relação ao relacionamento “Treina” a existência deste é sustentada e justificada pelos requisitos referentes à possibilidade do clientes participarem de aulas de grupo variadas e de treinos privados. De forma a distinguir se se trata de uma aula de grupo ou personalizada, podemos tanto olhar para a modalidade como para o espaço do ginásio que lhe está associado, sendo que um treino privado tem sempre como modalidade “Personal Training” e o espaço “0” associado e as aulas de grupo têm

sempre associadas uma modalidade diferente de “Personal Training” e um espaço que pode ter associado um número entre 1 e 5.

Em relação às restantes entidades e relacionamentos, a sua criação resume-se quase a uma tradução direta dos requisitos para o modelo conceptual.

Após a conclusão do esquema conceptual, os responsáveis pelo desenho do mesmo convocaram uma nova reunião entre toda a equipa envolvida no projeto. Cada interveniente, individualmente, analisou cuidadosamente cada elemento envolvido no esquema, e confirmou a correta transposição dos requisitos para o modelo, e vice-versa. Cruzando as considerações dos envolvidos, corrigiram alguns pormenores pontuais e aprovaram o modelo, tendo o projeto passado à próxima fase.

4. Modelação Lógica

4.1. Construção e Validação do Modelo de Dados Lógico

O modelo lógico foi construído a partir do nosso modelo conceptual, neste modelo as entidades e os seus relacionamentos são convertidos em tabelas e os identificadores de uma entidade passam a designar-se por Primary Key. Por sua vez, as Foreign Key correspondem a atributos ou conjunto de atributos que remetem para a Primary Key de outra tabela.

1. Funcionário:

1.1. Primary Key: idFuncionário como INT

1.2. Atributos: Nome: VARCHAR(45), NIF: INT, Sexo: TINYINT,
DataNascimento: Date, Email: VARCHAR(150), Rua: VARCHAR(100),
Cidade: VARCHAR(50), CódigoPostal: VARCHAR(50),
Distrito: VARCHAR(30), DataContratação: DATE, Ginasio: VARCHAR(45)
SalarioBase: VARCHAR(45), HorarioTrabalho: INT

1.3. Foreign Key: Não possui

2. FuncionarioTelefones:

2.1. Primary Key: Telemóvel como INT

2.2. Atributos: Não possui

2.3. Foreign Key: Funcionario: INT

3. FuncionarioFuncao:

3.1. Primary Key: Não possui

3.2. Atributos: Não possui

3.3. Foreign Key: Funcionario: INT, Funcao: VARCHAR(45)

4. Funcao:

4.1. Primary Key: Nome: VARCHAR(45)

4.2. Atributos: Não possui

4.3. Foreign Key: Não possui

5. FuncionarioGinasio:

5.1. Primary Key: DataPagamento: DATE

5.2. Atributos: SalarioTotal: DECIMAL(8,4), SalarioBase: DECIMAL(8,4),
HorarioTrabalho: INT

5.3. Foreign Key: Funcionario: INT, Ginasio: INT

6. Ginasio:

6.1. Primary Key: idGinasio: INT

6.2. Atributos: Nome: VARCHAR(30), DataAbertura: DATE, Rua: VARCHAR(100), CodigoPostal: VARCHAR(30), Cidade: VARCHAR(50), Distrito: VARCHAR(30)

6.3. Foreign Key: Não possui

7. FuncionarioCliente:

7.1. Primary Key: Id: VARCHAR(45)

7.2. Atributos: HorarioInicio: DATETIME, HorarioFim: DATETIME, Espaço: VARCHAR(30), Modalidade: VARCHAR(30), Valor: DECIMAL(8,4)

7.3. Foreign Key: Não possui

8. Cliente:

8.1. Primary Key: idCliente: INT

8.2. Atributos: NIF: INT, DataNascimento: DATE, Email: VARCHAR(150), Sexo: TINYINT, Nome: VARCHAR(45), CodigoPostal: VARCHAR(30), Rua: VARCHAR(100), Cidade: VARCHAR(50), Distrito: VARCHAR(30), CondicoesMedicas: VARCHAR(500), Extras: INT

8.3. Foreign Key: Não possui

9.GinasioCliente:

9.1. Primary Key: Não possui

9.2. Atributos: DataInscricao: DATE

9.3. Foreign Key: Ginasio: INT, Cliente: INT

10.Fatura:

10.1. Primary Key: idFatura: INT

10.2. Atributos: MetodoPagamento: TINYINT, ValorMensalidade: DECIMAL(8,2), DataEmissao: DATE, PrecoTotal: DECIMAL(8,2), DataPagamento: DATE

10.3. Foreign Key: Não possui

11.PlanoTreino:

11.1. Primary Key: Id: INT

11.2. Atributos: Preco: DECIMAL(8,2), HorarioFim: DATE, HorarioInicio: DATE

11.3. Foreign Key: Não possui

12.EntradasSaídas:

12.1. Primary Key: Data: DATE, Horario: TIME

12.2. Atributos: EntradaSaida: TINYINT, Ginasio: TINYINT

12.3 Foreign Key: Cliente: INT

13.ClienteTelefones:

13.1. Primary Key: Telemovel: INT

13.2. Atributos:

13.3 Foreign Key: Cliente: INT

4.2. Normalização de Dados

A normalização é um conjunto de regras (1FN, 2FN, 3FN), que pretende evitar a redundância de dados e aumentar o desempenho do modelo. Estas regras consistem na examinação de atributos de uma entidade e das relações entre entidades e têm como principal objetivo evitar anomalias na inclusão, exclusão e alteração de registos.

Para verificarmos a normalização do modelo temos que verificar se o mesmo satisfaz as 3 fórmulas normais:

- Primeira Fórmula Normal – A 1FN diz-nos que os atributos precisam de ser atómicos, ou seja, as tabelas não podem ter valores repetidos nem pode existir atributos multivalorados.
- Segunda Fórmula Normal – A 2FN diz-nos que só pode ser cumprida se a 1FN for satisfeita, e que os atributos normais devem depender unicamente da chave primária da tabela. Devemos eliminar as dependências parciais.
- Terceira Fórmula Normal – A 3FN diz-nos que após serem válidas as duas fórmulas anteriores, devemos verificar que todos os atributos de uma tabela devem ser independentes uns dos outros. É necessário eliminar as dependências transitivas.

Em relação ao nosso esquema lógico podemos afirmar que este se encontra normalizado apenas até à primeira forma, pois não grupos de dados repetidos, os valores de cada um dos atributos de uma tabela são todos atributos atómicos e cada tabela tem uma chave primária. Não podemos, contudo, afirmar que este se encontra numa forma de normalização superior, pois detetamos que pelo menos uma das tabelas não verifica a 2FN, não podendo o esquema estar na segunda forma normalizada. A tabela referida é a tabela FuncionarioGinasio e esta não respeita a segunda forma normal, pois apresenta uma dependência parcial, como verificamos de seguida:

- {DataPagamento, Funcionario, Ginasio} -> SalarioTotal, SalarioBase, HorarioTrabalho
- {DataPagamento, Funcionario} -> SalarioBase

Assim, cocluímos que, de facto, o esquema de encontra na primeria froma normalizada.

4.3. Apresentação e Explicação do Modelo Lógico Produzido

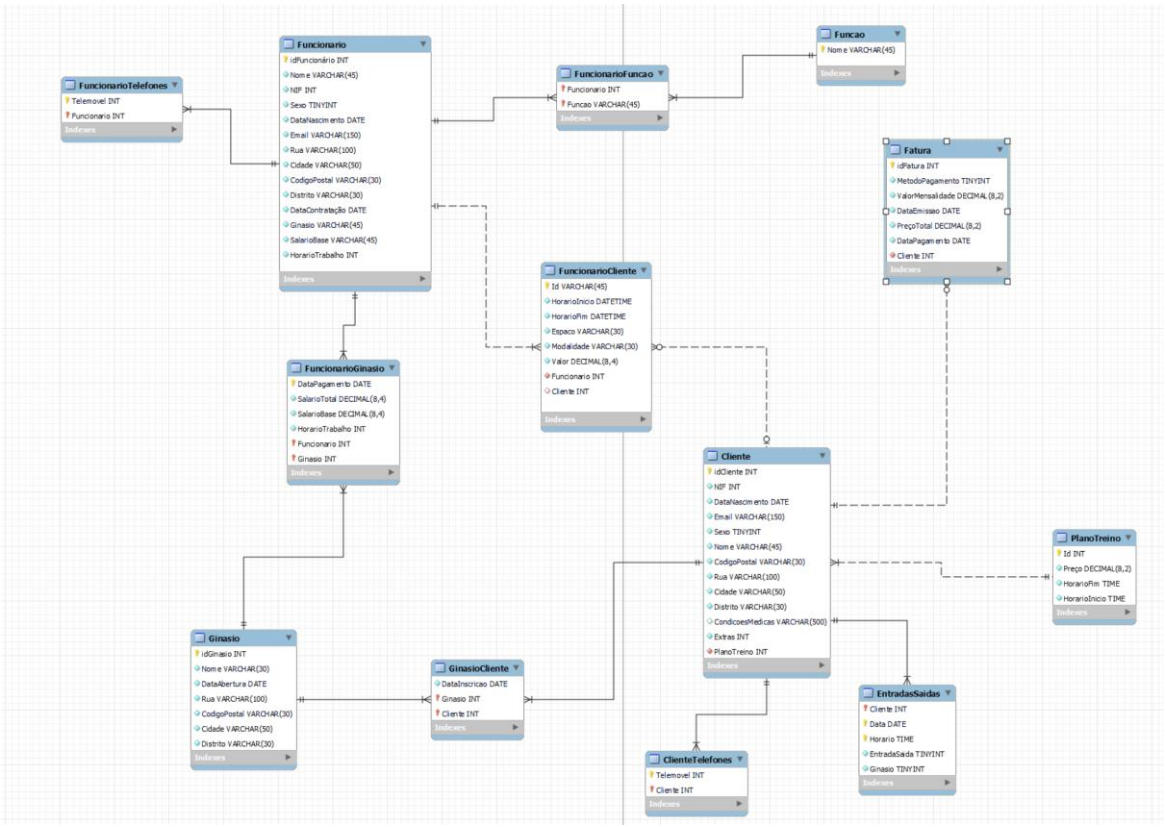


Figura 2 – Esquema Lógico.

Como podemos observar na imagem acima, o modelo lógico produzido possuí 13 tabelas.

A explicação deste é muito semelhante à do esquema conceptual pois a tradução do esquema conceptual para o lógico foi bastante direta, sendo a maior diferença o surgimento de tabelas no lugar dos relacionamentos e a adição de tabelas entre relacionamentos de cardinalidade N:M de forma a transformá-los em dois relacionamentos de cardinalidade 1: N. Observámos este surgimento de tabelas em relacionamentos N:M, por exemplo, na tabela FuncionarioFuncao aonde são guardados os registos que associam aos funcionários as funções que possuem.

Por sua vez, em relação à substituição dos restantes relacionamentos por tabelas, temos os exemplos das tabelas FuncionarioFuncao e FuncionarioCliente, que foram mencionadas aquando da descrição do modelo conceptual, sendo estas, respetivamente, responsáveis por guardar os registos das faturas e das aulas de grupo e aulas privadas dos clientes. Podemos ainda referir a tabela GinasioCliente, que armazena os registos relacionados com os clientes que

estão inscritos em cada ginásio, podendo um cliente estar inscrito em mais que um, desde que esse ginásio exista.

Importante realçar, que os atributos multivalorados, como os números de telemóveis dos clientes e funcionários e o registo de entradas e saídas dos clientes deram origem também a tabelas.

4.4. Validação do Modelo com Interrogações do Utilizador

Observando os requisitos recolhidos junto do Senhor Júnior, podemos aprovar que todos os requisitos são devidamente cumpridos:

- O ginásio é caracterizado por um id único, um nome, uma morada e uma data de abertura.
- Cada ginásio tem vários funcionários, que podem ter diversas funções;
- Existem apenas 5 funções possíveis, sendo estas: treinadores de sala, treinadores privados, treinadores de aulas de grupo, rececionistas e diretor.
- Um funcionário pode ter mais que uma função, mas não são possíveis todas as combinações de funções;
- Cada funcionário deve ter um ID único;
- Cada funcionário tem a si associadas as informações do seu nome, sexo (0 -> feminino; 1 -> Masculino), data de nascimento, morada, número de telemóvel, IBAN, NIF, e-mail e data de contratação;
- Cada funcionário tem ainda associado a si um ginásio onde trabalha, um salário base e um horário de trabalho.
- As moradas, são definidas pela rua, código postal, cidade e distrito;
- Cada ginásio tem vários clientes;
- Os clientes têm um número de identificação único, de forma a identificá-los univocamente;
- Em relação a cada cliente é necessário guardar informações do seu nome, sexo (0 -> feminino; 1 -> Masculino), data de nascimento, morada, número de telemóvel, IBAN, NIF, email e data de contratação;
- É necessário guardar a data de inscrição de cada cliente;
- Em relação ao cliente é ainda guardada, se necessário, alguma informação relevante sobre condições médicas;
- Cada cliente tem a si associado um plano de treino;
- Em relação aos clientes, é sempre registada a data e hora que entram e saem do ginásio e em que ginásio isso ocorreu;
- Cada cliente pode escolher ter 2 refills de água de 500 ml por treino, se pagar mais 1,25€ por mês. Caso contrário, apenas pode encher a garrafa de água na casa de banho;
- Cada cliente que tenha um plano de treino pode usufruir de todo o espaço sem custos adicionais, à exceção de aulas de grupo e aulas personalizadas;
- Os clientes se pretenderem ter a possibilidade de tomar banho, têm de pagar mais 4€ por mês;
- Existem 3 planos de treino e são caracterizados por um nome, pelo seu preço e pelo horário que têm associado;
- Os horários associados aos planos de treino são iguais todos os dias;
- Os planos de treino disponíveis são, o low-cost (7:00 - 13:00), o clássico (13:00-19:00) e o livre (7:00 - 23:00);
- Cada plano de treino inclui apenas a possibilidade de treinar com os professores de sala desse mesmo horário, sendo as aulas de grupo e personal training pagas à parte;
- Cada professor de sala tem um horário semanal fixo, sendo igual todos os dias da semana;
- Os horários possíveis são 3, sendo o das 7:00 às 13:00 designado por 1, o das 13:00 às 18:00 por 2 e o das 18:00 às 23:00 por 3.
- Os professores de sala para além de fazerem sala, podem ainda fazer treino especializado e/ou aulas de grupo, mas somente se tiverem essas especializações;

- Cada aula privada tem a si associada um funcionário com essa função, um cliente, um horário em que começa, um horário em que termina e o seu custo que varia com a duração da mesma;
- As aulas privadas decorrem sempre na sala de musculação e têm como modalidade "Personal Training";
- Cada aula de grupo tem a si associada a sua modalidade, um espaço, um horário de começo, um horário de término, um grupo de clientes e um funcionário com especialidade de professor de grupo;
- Um espaço é caracterizado por um número de 1 a 5, correspondendo a uma das 5 salas reservadas a aulas de grupo que o ginásio possuí;
- O espaço número 0 corresponde à sala de musculação.
- No final de cada mês é automaticamente emitida para cada cliente uma fatura que contém o valor total a ser pago por cada cliente, considerado o seu plano de treino, os extras que o ginásio oferece e as opcionais aulas de grupo e personal training;
- Os métodos de pagamento aceites são dinheiro vivo, multibanco e MBWay, representados, respetivamento, por 0, 1 e 2;
- Visto que são registadas as entradas e saídas dos clientes, estas são representadas, respetivamente, por um 0 e um 1;
- A fatura contém ainda um id único, a data de emissão da mesma, a data em que o cliente pagou e o método de pagamento;
- Em relação aos extras que cada cliente tem direito, caso este não opte por nenhum extra, é lhe associado o número 0, se optar somente por água o número 1, somente por banho o número 2 e se optar por ambos o número 3;
- Os treinadores recebem 25% do valor das aulas de grupo ou privadas, sendo este valor adicionado ao seu salário que irão receber no final do respetivo mês.
- O diretor do ginásio é o Senhor Júnior.

Todos os requisitos foram cumpridos devidamente.

5. Implementação Física

5.1. Tradução do Esquema Lógico Para o Sistema de Gestão de Bases de Dados

Passamos agora à tradução do esquema lógico previamente desenvolvido para o nosso Sistema de Gestão de Base de Dados (SGBD) escolhido, neste caso iremos usar o MySQL. Esta tradução é necessária porque compete ao SGBD facilitar o armazenamento, organização, manipulação e recuperação eficiente de dados, desvendando a complexidade subjacente dos dados, permitindo ao utilizador interagir na base de dados de forma mais simples e intuitiva.

```
-- -----------------------------------------------------
-- Schema Ginasio
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS Ginasio ;

-- -----------------------------------------------------
-- Schema Ginasio
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS Ginasio;
USE Ginasio ;
```

Figura 3 - Criação da SCHEMA.

Como podemos observar começamos com a criação da SCHEMA Ginásio, e passamos de seguida à criação das tabelas:

```
-- -----------------------------------------------------
-- Table PlanoTreino
-- -----------------------------------------------------
DROP TABLE IF EXISTS PlanoTreino ;

CREATE TABLE IF NOT EXISTS PlanoTreino (
  Id ENUM('1','2','3') NOT NULL,
  Preço DECIMAL(8,2) NOT NULL,
  HorarioFim TIME NOT NULL,
  HorarioInicio TIME NOT NULL,
  PRIMARY KEY (Id))
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 4 - Criação da Tabela PlanoTreino.

```
-- -----------------------------------------------------
-- Table Cliente
-- -----------------------------------------------------
DROP TABLE IF EXISTS Cliente ;

CREATE TABLE IF NOT EXISTS Cliente (
  idCliente INT UNSIGNED NOT NULL AUTO_INCREMENT,
  NIF INT NOT NULL,
  DataNascimento DATE NOT NULL,
  Email VARCHAR(150) NOT NULL,
  Sexo ENUM('0','1') NOT NULL,
  Nome VARCHAR(45) NOT NULL,
  CodigoPostal VARCHAR(30) NOT NULL,
  Rua VARCHAR(100) NOT NULL,
  Cidade VARCHAR(50) NOT NULL,
  Distrito VARCHAR(30) NOT NULL,
  CondicoesMedicas VARCHAR(500) NULL,
  Extras ENUM('0','1','2','3') DEFAULT '0',
  PlanoTreino ENUM('1','2','3') NOT NULL,
  PRIMARY KEY (idCliente),
  CONSTRAINT fk_Cliente_PlanoTreino
  FOREIGN KEY (PlanoTreino)
    REFERENCES PlanoTreino (Id)
    ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 5 - Criação da Tabela Cliente.

```
-- -----------------------------------------------------
-- Table ClienteTelefones
-- -----------------------------------------------------
DROP TABLE IF EXISTS ClienteTelefones ;

CREATE TABLE IF NOT EXISTS ClienteTelefones (
  Telemovel INT NOT NULL,
  Cliente INT UNSIGNED NOT NULL,
  PRIMARY KEY (Telemovel, Cliente),
  CONSTRAINT fk_ClienteTelefones_Cliente
  FOREIGN KEY (Cliente)
    REFERENCES Cliente (idCliente)
    ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 6 - Criação da tabela ClienteTelefones.

```sql
-- -----------------------------------------------------
-- Table EntradasSaidas
-- -----------------------------------------------------
DROP TABLE IF EXISTS EntradasSaidas ;

CREATE TABLE IF NOT EXISTS EntradasSaidas (
  Cliente INT UNSIGNED NOT NULL,
  Data DATE NOT NULL,
  Horario TIME NOT NULL,
  EntradaSaida ENUM('0','1') NOT NULL,
  Ginasio INT NOT NULL,
  PRIMARY KEY (Cliente, Data, Horario, Ginasio),
  CONSTRAINT fk_EntradasSaidas_Cliente
  FOREIGN KEY (Cliente)
    REFERENCES Cliente (idCliente)
    ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 7 - Criação da tabela EntradasSaidas

```
-- -----------------------------------------------------
-- Table Fatura
-- -----------------------------------------------------
DROP TABLE IF EXISTS Fatura ;

CREATE TABLE IF NOT EXISTS Fatura (
  idFatura INT UNSIGNED NOT NULL AUTO_INCREMENT,
  MetodoPagamento ENUM('0','1','2') NULL DEFAULT NULL,
  ValorMensalidade DECIMAL(8,2) NOT NULL,
  DataEmissao DATE NOT NULL,
  PreçoTotal DECIMAL(8,2) NOT NULL,
  DataPagamento DATE NULL DEFAULT NULL,
  Cliente INT NOT NULL,
  PRIMARY KEY (idFatura))
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 8 - Criação da tabela Fatura.

```
-- -----------------------------------------------------
-- Table Funcao
-- -----------------------------------------------------
DROP TABLE IF EXISTS Funcao ;

CREATE TABLE IF NOT EXISTS Funcao (
  Nome ENUM('Treinador de Sala', 'Treinador Privado', 'Treinador de Grupo', 'Rececionista', 'Diretor') NOT NULL,
  PRIMARY KEY (Nome))
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 9 - Criação da tabela Funcao.

```sql
-- -----------------------------------------------------
-- Table Funcionario
-- -----------------------------------------------------
DROP TABLE IF EXISTS Funcionario ;

CREATE TABLE IF NOT EXISTS Funcionario (
  idFuncionário INT UNSIGNED NOT NULL AUTO_INCREMENT,
  Nome VARCHAR(45) NOT NULL,
  NIF INT NOT NULL,
  Sexo ENUM('0','1') NOT NULL,
  DataNascimento DATE NOT NULL,
  Email VARCHAR(150) NOT NULL,
  Rua VARCHAR(100) NOT NULL,
  Cidade VARCHAR(50) NOT NULL,
  CodigoPostal VARCHAR(30) NOT NULL,
  Distrito VARCHAR(30) NOT NULL,
  DataContratação DATE NOT NULL,
  Ginasio INT NOT NULL,
  SalarioBase DECIMAL(8,4) NOT NULL,
  HorarioTrabalho ENUM('1','2','3') NOT NULL,
  PRIMARY KEY (idFuncionário))
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 10 – Criação da tabela Funcionario

```sql
-- -----------------------------------------------------
-- Table FuncionarioCliente
-- -----------------------------------------------------
DROP TABLE IF EXISTS FuncionarioCliente ;

CREATE TABLE IF NOT EXISTS FuncionarioCliente (
  idAula INT UNSIGNED NOT NULL AUTO_INCREMENT,
  HorarioInicio DATETIME NOT NULL,
  HorarioFim DATETIME NOT NULL,
  Espaco ENUM('0','1','2','3','4','5') DEFAULT '0',
  Modalidade VARCHAR(30) NOT NULL,
  Valor DECIMAL(8,4) NOT NULL,
  Funcionario INT UNSIGNED NOT NULL,
  Cliente INT UNSIGNED NULL,
  Ginasio INT UNSIGNED NOT NULL,
  PRIMARY KEY (idAula),
  CONSTRAINT fk_FuncionarioCliente_Cliente
  FOREIGN KEY (Cliente)
    REFERENCES Cliente (idCliente)
    ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_FuncionarioCliente_Funcionario
  FOREIGN KEY (Funcionario)
    REFERENCES Funcionario (idFuncionário)
    ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 11 - Criação da tabela FuncionarioCliente

```sql
-- -----------------------------------------------------
-- Table FuncionarioFuncao
-- -----------------------------------------------------
DROP TABLE IF EXISTS FuncionarioFuncao ;

CREATE TABLE IF NOT EXISTS FuncionarioFuncao (
  Funcionario INT UNSIGNED NOT NULL,
  Funcao ENUM('Treinador de Sala', 'Treinador Privado', 'Treinador de Grupo', 'Rececionista', 'Diretor') NOT NULL,
  PRIMARY KEY (Funcionario, Funcao),
  CONSTRAINT fk_FuncionarioFuncao_Funcao
  FOREIGN KEY (Funcao)
    REFERENCES Funcao (Nome)
    ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_FuncionarioFuncao_Funcionario
  FOREIGN KEY (Funcionario)
    REFERENCES Funcionario (idFuncionário)
    ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 12 - Criação da tabela FuncionarioFuncao

```
-- -----------------------------------------------------
-- Table FuncionarioGinasio
-- -----------------------------------------------------
DROP TABLE IF EXISTS FuncionarioGinasio ;

CREATE TABLE IF NOT EXISTS FuncionarioGinasio (
  DataPagamento DATE NOT NULL,
  SalarioBase DECIMAL(8,4) NOT NULL,
  SalarioTotal DECIMAL(8,4) NOT NULL,
  HorarioTrabalho ENUM('1','2','3') NOT NULL,
  Funcionario INT UNSIGNED NOT NULL,
  Ginasio INT UNSIGNED NOT NULL,
  PRIMARY KEY (DataPagamento, Funcionario, Ginasio),
  CONSTRAINT fk_FuncionarioGinasio_Funcionario
  FOREIGN KEY (Funcionario)
    REFERENCES Funcionario (idFuncionário)
    ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_FuncionarioGinasio_Ginasio
  FOREIGN KEY (Ginasio)
    REFERENCES Ginasio (idGinasio)
    ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 13 - Criação da tabela FuncionarioGinasio

```
-- -----------------------------------------------------
-- Table FuncionarioTelefones
-- -----------------------------------------------------
DROP TABLE IF EXISTS FuncionarioTelefones ;

CREATE TABLE IF NOT EXISTS FuncionarioTelefones (
  Telemovel INT NOT NULL,
  Funcionario INT UNSIGNED NOT NULL,
  PRIMARY KEY (Telemovel, Funcionario),
  CONSTRAINT fk_FuncionarioTelefones_Funcionario
  FOREIGN KEY (Funcionario)
    REFERENCES Funcionario (idFuncionário)
    ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 14 - Criação da tabela FuncionarioTelefones

```
-- -----------------------------------------------------
-- Table Ginasio
-- -----------------------------------------------------
DROP TABLE IF EXISTS Ginasio ;

CREATE TABLE IF NOT EXISTS Ginasio (
  idGinasio INT UNSIGNED NOT NULL AUTO_INCREMENT,
  Nome VARCHAR(30) NOT NULL,
  DataAbertura DATE NOT NULL,
  Rua VARCHAR(100) NOT NULL,
  CodigoPostal VARCHAR(30) NOT NULL,
  Cidade VARCHAR(50) NOT NULL,
  Distrito VARCHAR(30) NOT NULL,
  PRIMARY KEY (idGinasio))
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 15 - Criação da tabela Ginasio

```
-- -----------------------------------------------------
-- Table GinasioCliente
-- -----------------------------------------------------
DROP TABLE IF EXISTS GinasioCliente ;

CREATE TABLE IF NOT EXISTS GinasioCliente (
  DataInscricao DATE NOT NULL,
  Ginasio INT UNSIGNED NOT NULL,
  Cliente INT UNSIGNED NOT NULL,
  PRIMARY KEY (Ginasio, Cliente),
  CONSTRAINT fk_GinasioCliente_Ginasio
  FOREIGN KEY (Ginasio)
    REFERENCES Ginasio (idGinasio)
    ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_GinasioCliente_Cliente
  FOREIGN KEY (Cliente)
    REFERENCES Cliente (idCliente)
    ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARSET=utf8MB4;
```

Figura 16 - Criação da tabela GinasioCliente

Neste processo de criação de tabelas é possivel observar que fazemos uso das seguintes cláusulas: *CONSTRAINT*, *ON DELETE RESTRICT* e *ON UPDATE CASCADE*, que consideramos que merecem justificação. Utilizamos *CONSTRAINT* nas chaves estrangeiras para garantir uma relação entre duas tabelas, em que os valores da chave estrangeira de um tabela devem corresponder aos valores da chave primária da outra. ON DELETE RESTRICT é utilizado para bloquearmos qualquer tentativa de exclusão de um registo numa tabela pai enquanto existirem registos filhos associados. Por fim a cláusula *ON UPDATE CASCADE* garante-nos que quando o valor da chave primária de uma tabela pai é atualizado todos os valores das chaves estrangeiras na tabela filho também serão atualizados.

5.2. Tradução das Interrogações do Utilizador para SQL – Alguns Exemplos

De forma a implementar os requisitos especificados implementamos as seguintes queries:

```sql
-- Saber quantos clientes se encontram no ginásio neste momento

SELECT COUNT(SubQ.Cliente)
    FROM (
        SELECT Cliente, MAX(CONCAT(Data, ' ', Horario)) AS Momento
        FROM EntradasSaidas
        GROUP BY Cliente
    ) AS SubQ
        INNER JOIN EntradasSaidas AS E
        ON SubQ.Cliente = E.Cliente
            AND SubQ.Momento = CONCAT(E.Data, ' ', E.Horario)
    WHERE E.EntradaSaida = '0';
```

Figura 17 – Querie em SQL que mostra quantos clientes se encontram atualmente no ginasio

O objetivo desta query é saber quantos clientes se encontram no ginásio num determinado momento. Através da tabela dos clientes, a subquery seleciona os clientes que se encontram no ginásio num dado momento. A condição E.EntradaSaida = 0 garante que só são contabilizados clientes que estão no ginásio.

```sql
-- Saber quantos clientes já facultaram aulas de grupo

SELECT COUNT(DISTINCT Cliente)
        FROM FuncionarioCliente
        WHERE Modalidade != 'Personal Training'
            AND Cliente IS NOT NULL;
```

Figura 18 – Querie em SQL que mostra os que clientes que participaram em aulas de grupo

O count é utilizado para contar o número de valores únicos na coluna cliente. Através da tabela FuncionarioCliente aplica-se a condição da modalidade ser diferente de Personal Training.

```sql
-- Aceder ao histórico de aulas de grupo e aulas privadas

DROP PROCEDURE IF EXISTS clienteAulasHistorico;

DELIMITER $$
CREATE PROCEDURE clienteAulasHistorico
    (IN idCliente INT)
BEGIN

    SELECT *
        FROM FuncionarioCLiente
        WHERE Cliente = idCliente
            AND HorarioInicio < NOW();

END $$

CALL clienteAulasHistorico('1');
```

Figura 18 – Querie em SQL para aceder ao historico de aulas através do id de um cliente

O DROP PROCEDURE garante que não existe nenhum procedimento com o mesmo nome antes de criar um novo procedimento. Através do id de cliente, excluímos aulas ainda não terminadas para estas não entrarem na contagem. O call clienteAulasHistorico('1') chama o procedimento clienteAulasHistorico com o argumento 1 que representa o parâmetro idCliente.

```sql
-- Aceder às aulas de grupo e aulas privadas agendadas por um cliente

DROP PROCEDURE IF EXISTS clienteAulasAgendadas;

DELIMITER $$
CREATE PROCEDURE clienteAulasAgendadas
    (IN idCliente INT)
BEGIN

    SELECT *
        FROM FuncionarioCLiente
        WHERE Cliente = idCliente
            AND HorarioInicio > NOW();

END $$

CALL clienteAulasAgendadas('8');
```

Figura 19 - Querie em SQL para aceder às aulas agendadas através do id de um cliente

A query apresentada cria um procedimento chamado clienteAulasAgendadas que recebe um input idCliente. O procedimento seleciona todos os dados da tabela FuncionarioCliente para os quais o idCliente coincide e o HorarioInicio é maior do que o momento atual.

```sql
-- Em cada momento, consultar quais os clientes que se encontram inscritos em cada aula de grupo agendadas

SELECT FC.Cliente, C.Nome, FC.HorarioInicio, FC.HorarioFim, FC.Funcionario, FC.Modalidade, FC.Ginasio
    FROM Cliente AS C
        INNER JOIN FuncionarioCliente AS FC
        ON C.idCliente = FC.Cliente
    WHERE HorarioInicio > NOW()
    AND FC.Modalidade != 'Personal Training'
    AND FC.Cliente IS NOT NULL;
```

Figura 20 - Querie em SQL para retornar informações de cada cliente inscrito em modalidades coletivas (ou seja, qualquer uma que não personal training)

Basicamente esta query apresenta uma lista de clientes inscritos numa determinada aula de grupo agendada. A estratégia utilizada foi selecionar os clientes cuja modalidade seja aulas de grupo e verificar quais desses clientes estão inscritos nessa aula.

```sql
-- Aceder ao histórico das horas de entrada e saída de um determinado cliente.

DROP PROCEDURE IF EXISTS GetEntradaSaidaByClienteId;

DELIMITER $$
CREATE PROCEDURE GetEntradaSaidaByClienteId
    (IN clienteId INT)
BEGIN
    SELECT *
    FROM EntradasSaidas
    WHERE Cliente = clienteId;
END $$

CALL GetEntradaSaidaByClienteId('1');
```

Figura 21 – Querie em SQL para aceder ao historico de entradas e saidas de um cliente através do seu id

A query apresentada apresenta o histórico das entradas e saídas de um determinado cliente. A implementação é muito simples. Basta procurar o clienteId na tabela de EntradasSaidas.

```sql
-- Obter a lista de clientes que estão a usufruir de um determinado plano de treino.

DROP PROCEDURE IF EXISTS GetClientesByPlanoTreino;

DELIMITER $$
CREATE PROCEDURE GetClientesByPlanoTreino
    (IN planoTreinoId ENUM('1', '2', '3'))
BEGIN
    SELECT *
    FROM Cliente
    WHERE PlanoTreino = planoTreinoId;
END $$

CALL GetClientesByPlanoTreino('2');
```

Figura 22 - Querie em SQL para ver todas as informações dos clientes que estão inscritos num dado plano de treino

A anterior query procura todos os clientes na tabela “cliente” em que o plano de treino tem o mesmo valor que é passado como argumento.

```sql
DELIMITER $$
CREATE PROCEDURE GenerateExtrasUsageReport()
BEGIN
    DECLARE totalUsers INT;
    DECLARE waterRefillUsers INT;
    DECLARE showerUsers INT;
    DECLARE bothExtrasUsers INT;
    DECLARE waterRefillPercentage DECIMAL(5, 2);
    DECLARE showerPercentage DECIMAL(5, 2);
    DECLARE bothExtrasPercentage DECIMAL(5, 2);

    -- Numero total de clientes
    SELECT COUNT(*) INTO totalUsers FROM Cliente;

    -- Numero total de clientes a usufruir de refill de água
    SELECT COUNT(*) INTO waterRefillUsers FROM Cliente WHERE Extras = '1' OR Extras = '3';

    -- Numero total de clientes que pagam para poder tomar banho
    SELECT COUNT(*) INTO showerUsers FROM Cliente WHERE Extras = '2' OR Extras = '3';

    -- Numero total de clientes que pagam para poder tomar banho e encher a água
    SELECT COUNT(*) INTO bothExtrasUsers FROM Cliente WHERE Extras = '3';

    -- Calcular as percentagens
    SET waterRefillPercentage = (waterRefillUsers / totalUsers) * 100;
    SET showerPercentage = (showerUsers / totalUsers) * 100;
    SET bothExtrasPercentage = (bothExtrasUsers / totalUsers) * 100;

    SELECT 'Extras' AS Description, 'Number of Users' AS Metric, 'Percentage (%)' AS Percentage
    UNION ALL
    SELECT 'Water Refill', waterRefillUsers, waterRefillPercentage
    UNION ALL
    SELECT 'Shower Facilities', showerUsers, showerPercentage
    UNION ALL
    SELECT 'Both Extras (Water Refill and Shower)', bothExtrasUsers, bothExtrasPercentage;
END $$
```

Figura 23 – Querie em SQL que retorna as métricas relativas a cada extra

Nesta query conta-se todos os utilzadores da tabela cliente em que o “extra” é uma das 3 opções possíveis (1,2 ou 3) e depois calcu-se baseado no número de utilizadores total a percentagem de utilizadores que aproveita desse extra.

```sql
-- Saber quantos novos clientes o ginásio angariou em cada mês

SELECT YEAR(DataInscricao) AS Ano, MONTH(DataInscricao) AS Mes, COUNT(*) AS NovosClientes
    FROM GinasioCliente
    GROUP BY YEAR(DataInscricao), MONTH(DataInscricao)
    ORDER BY YEAR(DataInscricao), MONTH(DataInscricao);
```

Figura 24 – Querie em SQL que retorna o número de clientes que o ginásio angariou em cada mês

Para a resolução desta procuram-se os dados na tabela “GinasioCliente” e depois agrupam-se as informações relativas à data de inscrição de forma a calcular a soma de clientes que se inscreveram nesse mês e ano.

```sql
-- Entender qual a especialidade que está a render mais lucro para o ginásio

SELECT Modalidade, SUM(Valor) AS TotalGanho
    FROM FuncionarioCliente
    GROUP BY Modalidade
    ORDER BY SUM(Valor) DESC;
```

Figura 25 – Querie em SQL que retorna todas as especialidades por ordem de faturação individual

Para esta query, acede-se à tabela de relação entre funcionário e cliente e agrupa-se registos pela modalidade e pelo total ganho, seguido de uma ordenação decrescente. Desta forma, o primeiro registo da tabela irá corresponder à modalidade que traz mais lucro para o ginásio.

```sql
-- No final de cada mês deve ser possível conhecer a lotação do ginásio em diferentes intervalos de tempo do dia.

DROP PROCEDURE IF EXISTS CalcularFrequenciaEntradasSaidas;

DELIMITER //

CREATE PROCEDURE CalcularFrequenciaEntradasSaidas(IN mes INT, IN ano INT)
BEGIN
  SELECT
    CASE
      WHEN TIME(Horario) >= '07:00:00' AND TIME(Horario) < '13:00:00' THEN '07:00:00 - 13:00:00'
      WHEN TIME(Horario) >= '13:00:00' AND TIME(Horario) < '19:00:00' THEN '13:00:00 - 19:00:00'
      WHEN TIME(Horario) >= '19:00:00' AND TIME(Horario) <= '23:59:59' THEN '19:00:00 - 23:59:59'
    END AS IntervaloHorario,
    COUNT(*) AS Frequencia
  FROM EntradasSaidas
  WHERE YEAR(Data) = ano
    AND MONTH(Data) = mes
    AND EntradaSaida = '0'
  GROUP BY IntervaloHorario;
END //

DELIMITER ;
```

Figura 26 – Querie em SQL que mostra a lotação do ginásio em diferentes intervalos de tempo do dia

Para esta última query, procura-se na tabela que guarda os registos das entradas e saídas os movimentos comparando a hora de entrada de cada cliente com um dos três intervalos (07:00:00 - 13:00:00, 13:00:00 - 19:00:00 e 19:00:00 - 23:59:59.

5.3. Cálculo do Espaço da Base de Dados (Inicial e Taxa de Crescimento Anual)

Depois de várias reuniões com o Senhor Júnior, a previsão foi unânime:

Período de 6 Meses:

- Crescimento de 50% dos clientes.
- Crescimento de 10% dos funcionários.

Período de 1 Ano:

- Crescimento de 75% dos clientes.
- Crescimento de 15% dos funcionários.

Período de 3 Anos:

- Crescimento de 110% dos clientes.
- Crescimento de 30% dos funcionários.

5.4. Procedimentos Implementados

Primeiramente, implementamos 9 procedimentos:

- Associar um novo ginásio a um funcionário.
- Associar uma nova função a um funcionário.
- Associar um novo salário base a um funcionário.
- Associar um novo horário de trabalho a um funcionário.
- Associar um novo plano de treino a um cliente.
- Reservar uma aula de grupo.
- Agendar uma aula de grupo para um cliente.
- Agendar uma aula privada para um cliente.
- Registar uma entrada/saída de um cliente no ginásio.

É importante referir que em todos os procedimentos, ou quase todos, é realizada uma validação dos dados a ver se estes são válidos para impedir a inserção de dados sem sentido na base de dados. Podemos observar isso logo no procedimento a seguir, que antes de alterar qualquer tabela verifica que dados como o id do Ginásio que está a ser inserido ou que o id do funcionário que está a ser inserido existem já na base de dados, não levando a um estado de inconsistência dos dados. Da mesma forma, mesmo os dados sendo validos, em muitos procedimentos, avaliamos ainda se faz sentido a alteração que o utilizador pretende introduzir na base de dados, mas isso será avaliado ao aprofundar cada um dos procedimentos.

Em cada procedimento é ainda, por norma, adicionado acima dele a linha “DROP PROCEDURE IF EXISTS [Nome_Procedimento]” dando a possibilidade ao utilizador de remover esse procedimento da base de dados.

Nem todos os procedimentos serão explicados, visto que são muito semelhantes e de compreensão direta, tendo em conta o referido nos restantes exemplos de procedimentos.

```sql
-- Associa um novo ginásio a um Funcionário

DELIMITER $$
CREATE PROCEDURE funcionarioNovoGinasio
    (IN idGinasioNew INT, IN IdFuncionarioAdd INT)
BEGIN

    START TRANSACTION;

    -- Verificar se os dados a introduzir são válidos
    IF NOT EXISTS (
        SELECT *
            FROM Ginasio
            WHERE idGinasio = idGinasioNew
    ) OR NOT EXISTS (
        SELECT *
            FROM Funcionario
            WHERE idFuncionário = IdFuncionarioAdd
    ) THEN
        ROLLBACK;
    END IF;

    UPDATE Funcionario
        SET Ginasio = idGinasioNew
        WHERE idFuncionário = IdFuncionarioAdd;

    COMMIT;
END $$

CALL funcionarioNovoGinasio('2', '6');
SELECT *
    FROM Funcionario;
```

Figura 27 - Associar um novo ginásio a um funcionário.

Relativamente a este primeiro, após a validação dos dados e, caso não haja dados inválidos, processegue-se à atualização da tabela Funcionario através do comando UPDATE restringindo a alteração apenas ao registo que contém o id do funcionario igual ao introduzido pelo utilizador (através da claúsula WHERE).

```sql
-- Associa uma nova função a um Funcionário

DELIMITER $$
CREATE PROCEDURE funcionarioNovaFuncao
    (IN NomeFuncaoAdd VARCHAR(45), IN IdFuncionarioAdd INT)
BEGIN

    START TRANSACTION;

    -- Verificar se os dados a introduzir são válidos
    IF NOT EXISTS (
        SELECT *
            FROM Funcao
            WHERE Nome = NomeFuncaoAdd
    ) OR NOT EXISTS (
        SELECT *
            FROM Funcionario
            WHERE idFuncionário = IdFuncionarioAdd
    ) THEN
        ROLLBACK;
    END IF;

    IF NOT EXISTS (
        SELECT *
        FROM FuncionarioFuncao
        WHERE Funcionario = IdFuncionarioAdd
            AND Funcao = NomeFuncaoAdd
    ) AND NomeFuncaoAdd != 'Diretor' AND NomeFuncaoAdd != 'Rececionista' THEN
        INSERT INTO FuncionarioFuncao
            (Funcionario, Funcao)
            VALUES
                (IdFuncionarioAdd, NomeFuncaoAdd);
    ELSE
        ROLLBACK;
    END IF;

    COMMIT;
END $$

CALL funcionarioNovaFuncao('Treinador Privado', '4');
```

Figura 28 - Associar uma nova função a um funcionário

Em relação a este segundo procedimento, da mesma forma começa por verificar se os dados são válidos. De seguida, como mencionado que poderia acontecer, verifica se a alteração que se pretende alterar é possível, sendo neste caso verificar se a nova função a introduzir corresponde à função diretor ou rececionista, isto porque, de forma a respeitar os requisitos, nenhum tipo de treinador se pode tornar um rececionista ou diretor. Por fim, se verificar todas as condições impostas, insere a nova função associada ao id do funcionário que a recebe na tabela FuncionarioFuncao através da função INSERT INTO.

```sql
DROP PROCEDURE IF EXISTS funcionarioNovoSalarioBase;

-- Associa um novo salário base a um Funcionário

DELIMITER $$
CREATE PROCEDURE funcionarioNovoSalarioBase
    (IN salarioBaseNew DECIMAL(8,4), IN IdFuncionarioAdd INT)
BEGIN

    START TRANSACTION;

    -- Verificar se os dados a introduzir são válidos
    IF NOT EXISTS (
        SELECT *
            FROM Funcionario
            WHERE idFuncionário = IdFuncionarioAdd
    ) THEN
        ROLLBACK;
    END IF;

    UPDATE Funcionario
        SET SalarioBase = salarioBaseNew
        WHERE idFuncionário = IdFuncionarioAdd;

    COMMIT;
END $$

CALL funcionarioNovoSalarioBase('2500.00', '6');
SELECT *
    FROM Funcionario;
```

Figura 29 - Associar um novo salário base a um funcionário

```sql
DROP PROCEDURE IF EXISTS funcionarioNovoHorarioTrabalho;

-- Associa um novo horário de trabalho a um Funcionário

DELIMITER $$
CREATE PROCEDURE funcionarioNovoHorarioTrabalho
    (IN HorarioTrabalhoNew ENUM('1','2','3'), IN IdFuncionarioAdd INT)
BEGIN

    START TRANSACTION;

    -- Verificar se os dados a introduzir são válidos
    IF NOT EXISTS (
        SELECT *
            FROM Funcionario
            WHERE idFuncionário = IdFuncionarioAdd
    ) THEN
        ROLLBACK;
    END IF;

    UPDATE Funcionario
        SET HorarioTrabalho = HorarioTrabalhoNew
        WHERE idFuncionário = IdFuncionarioAdd;

    COMMIT;
END $$

CALL funcionarioNovoHorarioTrabalho('1', '3');
SELECT *
    FROM Funcionario;
```

Figura 30 - Associar um novo horário de trabalho a um funcionário

```sql
DROP PROCEDURE IF EXISTS clienteNovoPlanoTreino;

-- Associa um novo plano de treino a um cliente

DELIMITER $$
CREATE PROCEDURE clienteNovoPlanoTreino
    (IN PlanoTreinoNew ENUM('1','2','3'), IN IdClienteAdd INT)
BEGIN

    START TRANSACTION;

    -- Verificar se os dados a introduzir são válidos
    IF NOT EXISTS (
        SELECT *
            FROM Cliente
            WHERE idCliente = IdClienteAdd
    ) THEN
        ROLLBACK;
    END IF;

    UPDATE Cliente
        SET PlanoTreino = PlanoTreinoNew
        WHERE idCliente = IdClienteAdd;

    COMMIT;
END $$

CALL clienteNovoPlanoTreino('2', '3');
SELECT *
    FROM Cliente;
```

Figura 31 - Associar um novo plano de treino a um cliente

```sql
DROP PROCEDURE IF EXISTS reservarAulaGrupo;

-- Reserva uma aula de Grupo

DELIMITER $$
CREATE PROCEDURE reservarAulaGrupo
    (IN IdFuncionarioAdd INT, IN HorarioInicioAdd DATETIME,
        IN HorarioFimAdd DATETIME, IN EspacoAdd ENUM('1','2','3','4','5'), IN ModalidadeAdd VARCHAR(30), IN ValorAdd DECIMAL(8,4), IN GinasioAdd INT, IN Lotacao INT)
BEGIN

    DECLARE Counter INT;
    SET Counter = 1;

    START TRANSACTION;

    -- Verificar se os dados existem
    IF NOT EXISTS (
        SELECT *
            FROM Ginasio
            WHERE idGinasio = GinasioAdd
    ) OR NOT EXISTS (
        SELECT *
            FROM Funcionario
            WHERE idFuncionário = IdFuncionarioAdd
    ) THEN
        ROLLBACK;
    END IF;
```

```
    -- Verificar se os dados são válidos
        IF EXISTS (
            SELECT *
                FROM FuncionarioCliente
                WHERE HorarioInicio < HorarioFimAdd
                    AND HorarioFim > HorarioInicioAdd
                    AND Espaco = EspacoAdd
                    AND Ginasio = GinasioAdd
        ) OR EXISTS (
            SELECT *
                FROM FuncionarioCliente
                WHERE HorarioInicio < HorarioFimAdd
                    AND HorarioFim > HorarioInicioAdd
                    AND Funcionario = IdFuncionarioAdd
        ) OR NOT EXISTS (
            SELECT *
                FROM FuncionarioFuncao
                WHERE Funcionario = IdFuncionarioAdd
                    AND Funcao = 'Treinador de Grupo'
        ) THEN
            ROLLBACK;
        ELSE
            WHILE Counter <= Lotacao DO

            INSERT INTO FuncionarioCliente
                (HorarioInicio, HorarioFim, Espaco, Modalidade, Valor, Funcionario, Cliente, Ginasio)
                VALUES
                    (HorarioInicioAdd, HorarioFimAdd, EspacoAdd, ModalidadeAdd, ValorAdd, IdFuncionarioAdd, NULL, GinasioAdd);

                SET Counter = Counter + 1;
            END WHILE;
        END IF;

    COMMIT;
END $$


CALL reservarAulaGrupo('1', '2023-01-10 08:00:00', '2023-01-10 09:00:00', '2', 'Ginástica', '2.50', '1', '5');
SELECT *
```

Figura 32 - Reservar uma aula de grupo

Em relação a este procedimento podemos observar que mais uma vez, este começa por validar os dados. Em seguida, visto que o objetivo do mesmo é reservar uma aula de grupo, de forma aos clientes poderem se inscrever na mesma, é verificado se o funcionário que lhe está associado tem disponibilidade, ou seja, se não tem outra aula dentro do intervalo de tempo da que se está a introduzir, se o espaço que lhe está a ser associado se irá encontrar igualmente livre durante toda a duração da aula e se, de facto, o funcionário que se está a tentar atribuir à aula tem a especialização necessária para o fazer. Se tudo isto for verificado, são inseridos tantos registos quanto o valor indicado pela lotação introduzida para a aula de grupo. Importante referir que estes são introduzidos com o valor do atributo cliente a NULL, de forma a dar a possibilidade de clientes se inscreverem.

```sql
-- Agenda uma aula de grupo para um cliente

DELIMITER $$
CREATE PROCEDURE marcaAulaGrupo
    (IN IdClienteAdd INT, IN HorarioInicioAdd DATETIME, IN HorarioFimAdd DATETIME, IN EspacoAdd ENUM('1','2','3','4','5'), IN GinasioAdd INT)
BEGIN

    DECLARE idVaga INT;

    START TRANSACTION;

    -- Verificar se existe alguma vaga disponível para essa aula
    SELECT idAula
    INTO idVaga
            FROM FuncionarioCliente
            WHERE Ginasio = GinasioAdd
                AND HorarioInicio = HorarioInicioAdd
                AND Espaco = EspacoAdd
                AND Cliente IS NULL
            ORDER BY idAula ASC
            LIMIT 1;

    IF idVaga IS NULL THEN
        ROLLBACK;
    ELSEIF NOT EXISTS (
        SELECT *
            FROM FuncionarioCliente
            WHERE HorarioInicio < HorarioFimAdd
                AND HorarioFim > HorarioInicioAdd
                AND Cliente = IdClienteAdd
    ) THEN
        UPDATE FuncionarioCliente
            SET Cliente = idClienteAdd
            WHERE idAula = idVaga;
    ELSE
        ROLLBACK;
    END IF;

    COMMIT;
END $$
```

Figura 33 - Agendar uma aula de grupo para um cliente

Este procedimento encontra-se relacionado com o anterior e é responsável por adicionar a uma aula de grupo um cliente, apenas se este já não tiver já uma atividade no período de duração da aula de grupo, se o mesmo não se tiver já inscrito e se a lotação máxima ainda não tiver sido atingida.

```sql
-- Agenda uma aula privada para um cliente

DELIMITER $$
CREATE PROCEDURE marcaAulaPrivada
    (IN idClienteAdd INT, IN idFuncionarioAdd INT, IN HorarioInicioAdd DATETIME, IN HorarioFimAdd DATETIME, IN ValorAdd DECIMAL(8,4), IN GinasioAdd INT)
BEGIN

    START TRANSACTION;

    -- Verificar se os dados existem
    IF NOT EXISTS (
        SELECT *
            FROM Ginasio
            WHERE idGinasio = GinasioAdd
    ) OR NOT EXISTS (
        SELECT *
            FROM Funcionario
            WHERE idFuncionário = IdFuncionarioAdd
    ) OR NOT EXISTS (
        SELECT *
            FROM Cliente
            WHERE idCliente = idClienteAdd
    ) THEN
        ROLLBACK;
    END IF;
```

```
    -- Verificar se os dados são válidos
    IF EXISTS (
        SELECT *
            FROM FuncionarioCliente
            WHERE HorarioInicio < HorarioFimAdd
                AND HorarioFim > HorarioInicioAdd
                AND Funcionario = idFuncionarioAdd
    ) OR EXISTS (
        SELECT *
            FROM FuncionarioCliente
            WHERE HorarioInicio < HorarioFimAdd
                AND HorarioFim > HorarioInicioAdd
                AND Cliente = idClienteAdd
    ) OR NOT EXISTS (
        SELECT *
            FROM FuncionarioFuncao
            WHERE Funcionario = IdFuncionarioAdd
                AND Funcao = 'Treinador Privado'
    ) THEN
        ROLLBACK;
    ELSE
        INSERT INTO FuncionarioCliente
            (HorarioInicio, HorarioFim, Espaco, Modalidade, Valor, Funcionario, Cliente, Ginasio)
            VALUES
                (HorarioInicioAdd, HorarioFimAdd, '0', 'Personal Training', ValorAdd, idFuncionarioAdd, idClienteAdd, GinasioAdd);
    END IF;

    COMMIT;
END $$


CALL marcaAulaPrivada('1', '2', '2023-01-10 10:00:00', '2023-01-10 11:00:00', '10.00', '1');
SELECT *
    FROM FuncionarioCliente;
```

Figura 34 - Agendar uma aula privada para um cliente

Este procedimento é responsável por reservar uma aula privada executando as mesmas verificações feitas ao criar uma aula de grupo, nomeadamente, se o funcionário tem a função necessária para o executar, se este tem disponibilidade e se o cliente também tem disponibilidade. Se isto for verificado, depois apenas se insere um registo na tabela FuncionarioCliente com toda a informação e o espaço associado “0”, correspondente à sala de musculação e a modalidade “Personal Training”.

```sql
CREATE PROCEDURE entradaSaidaClienteGinasio
    (IN idClienteAdd INT, DataAdd DATE, IN HorarioAdd TIME, IN GinasioAdd INT)
BEGIN

    DECLARE UltimoHorarioRegistado TIME;
    DECLARE UltimaDataRegistada DATE;
    DECLARE UltimoEntradaSaidaRegistada ENUM('0','1');

    START TRANSACTION;

    -- Verificar se os dados existem
    IF NOT EXISTS (
        SELECT *
            FROM Ginasio
            WHERE idGinasio = GinasioAdd
    ) OR NOT EXISTS (
        SELECT *
            FROM Cliente
            WHERE idCliente = idClienteAdd
    ) THEN
        ROLLBACK;
    END IF;

    SELECT Horario, Data, EntradaSaida
    INTO UltimoHorarioRegistado, UltimaDataRegistada, UltimoEntradaSaidaRegistada
    FROM EntradasSaidas
        WHERE Cliente = idClienteAdd
            AND Ginasio = GinasioAdd
        ORDER BY Data DESC
        LIMIT 1;

    IF UltimaDataRegistada > DataAdd OR
        (UltimaDataRegistada = DataAdd AND UltimoHorarioRegistado >= HorarioAdd
    ) THEN
        ROLLBACK;
    ELSEIF UltimoEntradaSaidaRegistada = '0' THEN
        INSERT INTO EntradasSaidas
            (Cliente, Data, Horario, EntradaSaida, Ginasio)
            VALUES
                (idClienteAdd, DataAdd, HorarioAdd, '1', GinasioAdd);
    ELSEIF UltimoEntradaSaidaRegistada = '1' OR UltimoEntradaSaidaRegistada IS NULL THEN
        INSERT INTO EntradasSaidas
            (Cliente, Data, Horario, EntradaSaida, Ginasio)
            VALUES
                (idClienteAdd, DataAdd, HorarioAdd, '0', GinasioAdd);
    END IF;

    COMMIT;
END $$
```

Figura 35 - Registar uma entrada/saída de um cliente no ginásio

Simplesmente determina se o cliente já se encontra no ginásio ou não e realiza a operação contrária, por exemplo, se o último registo associado a um cliente diz que ele entrou, então introduzimos um novo a dizer que saiu, e vice-versa.

5.5. TRIGGERS

Em relação aos Triggers, criamos somente dois para socorrer a execução de outros dois procedimentos que vamos já passar a apresentar.

```
DROP TRIGGER  IF EXISTS Funcionario_insert_trigger;

-- Envia todos os vencimentos em falta quando se tenta alterar o ginásio ou o horário ou o salário base associado a um funcionário, assegurando assim a fiabilidade e coerência temporal

DELIMITER $$
CREATE TRIGGER Funcionario_insert_trigger
BEFORE UPDATE ON Funcionario
FOR EACH ROW
BEGIN
    IF NEW.Ginasio <> OLD.Ginasio THEN
        CALL emiteSalarios();
    ELSEIF NEW.SalarioBase <> OLD.SalarioBase THEN
        CALL emiteSalarios();
    ELSEIF NEW.HorarioTrabalho <> OLD.HorarioTrabalho THEN
        CALL emiteSalarios();
    END IF;

END $$


DROP TRIGGER  IF EXISTS Cliente_insert_trigger;

-- Envia todos as faturas em falta quando se tenta alterar o plano de treino de um cliente, assegurando assim a fiabilidade e coerência temporal das faturas emitidas

DELIMITER $$
CREATE TRIGGER Cliente_insert_trigger
BEFORE UPDATE ON Cliente
FOR EACH ROW
BEGIN
    IF NEW.PlanoTreino <> OLD.PlanoTreino THEN
        CALL emiteFaturas();
    END IF;

END $$
```

Figura 36 – TRIGGERS

```
DROP PROCEDURE IF EXISTS emiteSalarios;

-- Emite os vencimentos para todos os funcionários

DELIMITER $$
CREATE PROCEDURE emiteSalarios ()
BEGIN

    DECLARE Counter INT;
    DECLARE MaxCounter INT;
    DECLARE DataContrataçãoNew DATE;
    DECLARE SalarioBaseNew DECIMAL(8,2);
    DECLARE SalarioTotalNew DECIMAL(8,2);
    DECLARE HorarioTrabalhoNew ENUM('1','2','3');
    DECLARE FuncionarioNew INT;
    DECLARE GinasioNew INT;


    -- Inserir os vencimentos em falta de todos os funcionários até ao momento na tabela FuncionarioGinasio
    SET Counter = 1;
    SELECT idFuncionário
    INTO MaxCounter
        FROM Funcionario
        ORDER BY idFuncionário DESC
        LIMIT 1;
```

```
WHILE Counter <= MaxCounter DO

    -- Calcular os dados correspondetes ao vencimento de cada funcionário

    SELECT Ginasio, SalarioBase, DataContratação, HorarioTrabalho
    INTO GinasioNew, SalarioBaseNew, DataContrataçãoNew, HorarioTrabalhoNew
        FROM Funcionario
        WHERE idFuncionário = Counter;


    -- Inserir os vencimentos todos em falta de um funcionário
    WHILE DataContrataçãoNew <= CURRENT_DATE() DO

        IF NOT EXISTS (
            SELECT *
                FROM FuncionarioGinasio
                WHERE DataPagamento = DataContrataçãoNew
                    AND Funcionario = Counter)
        THEN
            SELECT SUM(Aulas.Valor)
            INTO SalarioTotalNew
                FROM (
                    SELECT DISTINCT Funcionario, HorarioInicio, Valor
                        FROM FuncionarioCliente
                        WHERE HorarioInicio < DataContrataçãoNew
                            AND HorarioInicio >= DATE_SUB(DataContrataçãoNew, INTERVAL 1 MONTH)
                            AND Funcionario = Counter) AS Aulas
                GROUP BY Funcionario;

            IF SalarioTotalNew IS NULL OR SalarioTotalNew = 0 THEN
                SET SalarioTotalNew = SalarioBaseNew;
            ELSE
                SET SalarioTotalNew = SalarioTotalNew + SalarioBaseNew;
            END IF;

              INSERT INTO FuncionarioGinasio
                  (DataPagamento, SalarioBase, SalarioTotal, HorarioTrabalho, Funcionario, Ginasio)
                  VALUES
                      (DataContrataçãoNew, SalarioBaseNew, SalarioTotalNew, HorarioTrabalhoNew, Counter, GinasioNew);
          END IF;

          SET SalarioTotalNew = 0;
          SET DataContrataçãoNew = DATE_ADD(DataContrataçãoNew, INTERVAL 1 MONTH);
      END WHILE;

      SET Counter = Counter + 1;
  END WHILE;

END $$

CALL emiteSalarios();
SELECT *
    FROM FuncionarioGinasio;
```

Figura 37 – Procedimento que emite todos vencimentos em falta para todos os funcionários

```sql
-- Emite as faturas para todos os Clientes

DELIMITER $$
CREATE PROCEDURE emiteFaturas ()
BEGIN

    DECLARE Counter INT;
    DECLARE MaxCounter INT;
    DECLARE ValorMensalidadeNew DECIMAL(8,2);
    DECLARE DataEmissaoNew DATE;
    DECLARE PreçoTotalNew DECIMAL(8,2);
    DECLARE ClienteNew INT;
    DECLARE ValorExtrasCliente ENUM('0','1','2','3');


    -- Inserir as faturas todas em falta dos clientes até ao momento na tabela Fatura
    SET Counter = 1;
    SELECT idCliente
    INTO MaxCounter
        FROM Cliente
        ORDER BY idCliente DESC
        LIMIT 1;

    WHILE Counter <= MaxCounter DO

        -- Calcular os dados a inserir na nova fatura
        SELECT DataInscricao
        INTO DataEmissaoNew
            FROM GinasioCliente
            WHERE Cliente = Counter;

        SELECT P.Preço
        INTO ValorMensalidadeNew
            FROM PlanoTreino as P
                INNER JOIN Cliente as C
                ON C.PlanoTreino = P.Id
            WHERE C.idCliente = Counter;

        SELECT Extras
        INTO ValorExtrasCliente
            FROM Cliente
            WHERE idCliente = Counter;
```

```
        -- Inserir as faturas todas em falta de um cliente
        WHILE DataEmissaoNew <= CURRENT_DATE() DO

            IF NOT EXISTS (
                SELECT *
                    FROM Fatura
                    WHERE DataEmissao = DataEmissaoNew
                        AND Cliente = Counter)
            THEN
                SELECT SUM(Valor)
                INTO PreçoTotalNew
                    FROM FuncionarioCliente
                    WHERE HorarioInicio < DataEmissaoNew
                        AND HorarioInicio >= DATE_SUB(DataEmissaoNew, INTERVAL 1 MONTH)
                        AND Cliente = Counter
                    GROUP BY (Cliente);

                IF PreçoTotalNew IS NULL OR PreçoTotalNew = 0 THEN
                    SET PreçoTotalNew = ValorMensalidadeNew;
                ELSE
                    SET PreçoTotalNew = PreçoTotalNew + ValorMensalidadeNew;
                END IF;

                IF ValorExtrasCliente = '1' THEN
                    SET PreçoTotalNew = PreçoTotalNew + 1.25;
                ELSEIF ValorExtrasCliente = '2' THEN
                    SET PreçoTotalNew = PreçoTotalNew + 4.00;
                ELSEIF ValorExtrasCliente = '3' THEN
                    SET PreçoTotalNew = PreçoTotalNew + 5.25;
                END IF;

                INSERT INTO Fatura
                    (MetodoPagamento, ValorMensalidade, DataEmissao, PreçoTotal, DataPagamento, Cliente)
                    VALUES
                        (NULL, ValorMensalidadeNew, DataEmissaoNew, PreçoTotalNew, NULL, Counter);
            END IF;

            SET PreçoTotalNew = 0;
            SET DataEmissaoNew = DATE_ADD(DataEmissaoNew, INTERVAL 1 MONTH);
        END WHILE;

        SET Counter = Counter + 1;
    END WHILE;

END $$
```

Figura 38 - Procedimento que emite todas as faturas em falta para todos os clientes

```sql
DROP PROCEDURE IF EXISTS pagaFaturaCliente;

-- Paga uma fatura de um cliente

DELIMITER $$
CREATE PROCEDURE pagaFaturaCliente
    (IN idClienteAdd INT, IN MetodoPagamentoAdd ENUM('0','1','2'), IN DataPagamentoAdd DATE)
BEGIN

    DECLARE idFaturaPagar INT;

    SELECT idFatura
    INTO idFaturaPagar
        FROM Fatura as F
            INNER JOIN Cliente as C
            ON C.idCliente = F.Cliente
        WHERE F.DataEmissao <= DataPagamentoAdd
            AND C.idCliente = idClienteAdd
            AND F.MetodoPagamento IS NULL
        ORDER BY F.DataEmissao ASC
        LIMIT 1;

    UPDATE Fatura
        SET MetodoPagamento = MetodoPagamentoAdd,
            DataPagamento = DataPagamentoAdd
        WHERE idFatura = idFaturaPagar;

END $$

CALL pagaFaturaCliente('1', '1', '2023-03-01');
SELECT *
    FROM Fatura;
```

Figura 39 – Procedimento que paga uma fatura associada a um cliente

5.6. Plano de Segurança e Recuperação de Dados

A segurança é essencial em qualquer tipo de base de dados. Criamos um plano de segurança e recuperação de dados que inclui as seguintes componentes:

- **Backup regular dos dados**. Estabelecemos uma política de backup para realizar cópias regulares dos dados armazenados na base de dados.
- **Teste de restauração de backup**: Periodicamente, testamos a restauração dos backups para garantir que os dados possam ser recuperados com sucesso em caso de falha.
- **Controlo de acesso e autenticação**: Implementamos mecanismos de controlo de acesso e autenticação para garantir que apenas utilizadores autorizados possam aceder à base de dados.
- **Criptografia de dados**: Para proteger dados confidenciais.
- **Atualizações de segurança**: Mantemos o sistema de base de dados atualizado com as últimas atualizações de segurança disponíveis.

6.1. Povoamento das Estruturas de Dados Para Análise

O povoamento foi realizado ambas manualmente (num ficheiro *SQL* criado com esse propósito), quanto automaticamente, via um script escrito em *Phyton que se coneta à base de dados e executa comandos de forma a povoar as tabelas desejadas.*

No que toca à implementação manual, a mesma foi feita do seguinte modo:

```
-- Inserir dados na tabela PlanoTreino
INSERT INTO PlanoTreino (Id, Preço, HorarioFim, HorarioInicio)
VALUES
  ('1', '25.00', '7:00:00', '13:00:00'),
  ('2', '30.00', '7:00:00', '19:00:00'),
  ('3', '35.00', '7:00:00', '23:00:00');
```

Figura 40 - Povoamento da tabela PlanoTreino

```
-- Inserir dados na tabela Cliente
INSERT INTO Cliente (NIF, DataNascimento, Email, Sexo, Nome, CodigoPostal, Rua, Cidade, Distrito, CondicoesMedicas,
    Extras, PlanoTreino)
VALUES
  ('123456789', '1990-05-15', 'joao.silva@example.com', '1', 'João Silva', '12345', 'Rua dos Lírios', 'Lisboa',
    'Lisboa', 'Nenhuma', '0', '1'),
  ('987654321', '1985-08-25', 'maria.santos@example.com','0', 'Maria Santos', '54321', 'Rua da Fonte', 'Porto',
    'Porto', 'Nenhuma', '1', '2'),
  ('654321987', '1992-12-10', 'pedro.oliveira@example.com', '1', 'Pedro Oliveira', '67890', 'Rua dos Girassóis',
    'Braga', 'Braga', 'Nenhuma', '1', '3'),
  ('111222333', '1995-02-18', 'ana.rodrigues@example.com', '0', 'Ana Rodrigues', '55555', 'Rua das Oliveiras',
    'Coimbra', 'Coimbra', 'Nenhuma', '1', '1'),
  ('444555666', '1988-07-01', 'ricardo.faria@example.com', '1', 'Ricardo Faria', '99999', 'Rua dos Cravos', 'Faro',
    'Faro', 'Nenhuma', '2', '2'),
  ('777888999', '1993-09-22', 'carla.silveira@example.com', '0', 'Carla Silveira', '77777', 'Rua dos Pinheiros',
    'Viseu', 'Viseu', 'Nenhuma', '3', '3'),
  ('222333444', '1997-12-05', 'gustavo.pereira@example.com', '1', 'Gustavo Pereira', '44444', 'Rua dos Jasmins',
    'Évora', 'Évora', 'Nenhuma', '2', '1'),
  ('555666777', '1990-04-30', 'patricia.sousa@example.com', '0', 'Patrícia Sousa', '11111', 'Rua das Rosas',
    'Bragança', 'Bragança', 'Nenhuma', '2', '2'),
  ('888999000', '1987-11-13', 'miguel.carvalho@example.com', '1', 'Miguel Carvalho', '88888', 'Rua dos Crisântemos',
    'Guarda', 'Guarda', 'Nenhuma', '1', '3'),
  ('333444555', '1994-03-27', 'sara.gomes@example.com', '0', 'Sara Gomes', '22222', 'Rua das Camélias', 'Aveiro',
    'Aveiro', 'Nenhuma', '3', '1');
```

Figura 41 - Povoamento da tabela Cliente

```sql
-- Inserir dados na tabela ClienteTelefones
INSERT INTO ClienteTelefones (Telemovel, Cliente)
VALUES
  ('912345678', '1'),
  ('923456789', '1'),
  ('934567890', '2'),
  ('945678901', '3'),
  ('956789012', '4'),
  ('967890123', '4'),
  ('978901234', '5'),
  ('989012345', '6'),
  ('990123456', '7'),
  ('901234567', '8'),
  ('987678099', '9'),
  ('997637123', '10');
```

Figura 42 - Povoamento da tabela ClienteTelefones

```sql
-- Inserir dados na tabela EntradasSaidas
INSERT INTO EntradasSaidas (Cliente, Data, Horario, EntradaSaida, Ginasio)
VALUES
  ('1', '2023-01-01', '08:30:00', '0', '1'),
  ('1', '2023-01-01', '9:45:00', '1', '1'),
  ('2', '2023-01-01', '09:15:00', '0', '1'),
  ('2', '2023-01-01', '11:30:00', '1', '1'),
  ('3', '2023-01-01', '07:45:00', '0', '1'),
  ('3', '2023-01-01', '9:30:00', '1', '1'),
  ('4', '2023-01-01', '18:00:00', '0', '1'),
  ('4', '2023-01-01', '19:15:00', '1', '1'),
  ('5', '2023-01-01', '08:15:00', '0', '1'),
  ('5', '2023-01-01', '9:00:00', '1', '1'),
  ('6', '2023-01-01', '11:30:00', '0', '1'),
  ('6', '2023-01-02', '07:45:00', '1', '1'),
  ('6', '2023-01-02', '9:30:00', '0', '1'),
  ('6', '2023-01-04', '18:00:00', '1', '1'),
  ('9', '2023-01-05', '19:15:00', '0', '1'),
  ('9', '2023-01-05', '08:15:00', '1', '1'),
  ('10', '2023-01-06', '9:00:00', '0', '1');
```

Figura 4 - Povoamento da tabela EntradasSaidas

```sql
-- Inserir dados na tabela Fatura
INSERT INTO Fatura (MetodoPagamento, ValorMensalidade, DataEmissao, PreçoTotal, DataPagamento, Cliente)
VALUES
  (NULL, '25.00', '2023-01-10', '26.25', NULL, '1'),
  (NULL, '25.00', '2023-02-10', '28.75', NULL, '1'),
  (NULL, '30.00', '2023-01-15', '36.50', NULL, '2'),
  (NULL, '30.00', '2023-02-15', '34.00', NULL, '2'),
  (NULL, '25.00', '2023-01-01', '29.00', NULL, '4'),
  (NULL, '25.00', '2023-02-01', '37.50', NULL, '4'),
  (NULL, '25.00', '2023-03-01', '29.00', NULL, '4'),
  (NULL, '25.00', '2023-04-01', '29.00', NULL, '4'),
  (NULL, '30.00', '2023-01-12', '35.25', NULL, '5'),
  (NULL, '30.00', '2023-02-12', '45.00', NULL, '5'),
  (NULL, '30.00', '2023-03-12', '35.25', NULL, '5'),
  (NULL, '35.00', '2023-01-15', '35.00', NULL, '6'),
  (NULL, '25.00', '2023-01-02', '30.25', NULL, '7'),
  (NULL, '25.00', '2023-02-02', '45.75', NULL, '7'),
  (NULL, '30.00', '2023-01-05', '35.25', NULL, '8'),
  (NULL, '30.00', '2023-02-05', '66.25', NULL, '8'),
  (NULL, '30.00', '2023-03-05', '35.25', NULL, '8'),
  (NULL, '25.00', '2023-01-19', '25.00', NULL, '10');
```

Figura 44 - Povoamento da tabela Fatura

```sql
-- Inserir dados na tabela Funcao
INSERT INTO Funcao (Nome)
VALUES
  ('Treinador de Sala'),
  ('Treinador Privado'),
  ('Treinador de Grupo'),
  ('Rececionista'),
  ('Diretor');
```

Figura 44 - Povoamento da tabela Funcao

```sql
-- Inserir dados na tabela Funcionario
INSERT INTO Funcionario (Nome, NIF, Sexo, DataNascimento, Email, Rua, Cidade, CodigoPostal, Distrito, DataContratação,
    Ginasio, SalarioBase, HorarioTrabalho)
VALUES
  ('Júnior Neto', '123456789', '1', '1990-05-15', 'joao.silva@example.com', 'Rua dos Lírios', 'Lisboa', '12345',
    'Lisboa', '2022-01-01', '1', '1500.00', '1'),
  ('Maria Santos', '987654321', '0', '1985-08-25', 'maria.santos@example.com', 'Rua da Fonte', 'Porto', '54321',
    'Porto', '2022-02-01', '1', '1200.00', '2'),
  ('Pedro Oliveira', '654321987', '1', '1992-12-10', 'pedro.oliveira@example.com', 'Rua dos Girassóis', 'Braga',
    '67890', 'Braga', '2022-03-01', '1', '900.00', '3'),
  ('Ana Rodrigues', '111222333', '0', '1995-02-18', 'ana.rodrigues@example.com', 'Rua das Oliveiras', 'Coimbra',
    '55555', 'Coimbra', '2022-04-01', '1', '1100.00', '1'),
  ('Ricardo Faria', '444555666', '1', '1988-07-01', 'ricardo.faria@example.com', 'Rua dos Cravos', 'Faro', '99999',
    'Faro', '2022-05-01', '1', '1000.00', '2'),
  ('Carla Silveira', '777888999', '0', '1993-09-22', 'carla.silveira@example.com', 'Rua dos Pinheiros', 'Viseu',
    '77777', 'Viseu', '2022-06-01', '1', '1100.00', '1');
```

Figura 45 - Povoamento da tabela Funcionario

```sql
-- Inserir dados na tabela FuncionarioFuncao
INSERT INTO FuncionarioFuncao (Funcionario, Funcao)
VALUES
  ('1', 'Diretor'),
  ('2', 'Rececionista'),
  ('3', 'Treinador de Sala'),
  ('3', 'Treinador de Grupo'),
  ('3', 'Treinador Privado'),
  ('4', 'Treinador de Grupo'),
  ('5', 'Treinador de Sala'),
  ('5', 'Treinador de Grupo'),
  ('6', 'Treinador de Grupo'),
  ('6', 'Treinador Privado');
```

Figura 46 - Povoamento da tabela FuncionarioFuncao

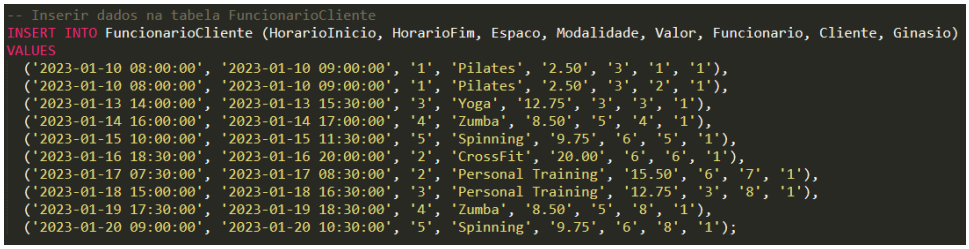
```sql
-- Inserir dados na tabela FuncionarioFuncao
INSERT INTO FuncionarioFuncao (Funcionario, Funcao)
VALUES
  ('1', 'Diretor'),
  ('2', 'Rececionista'),
  ('3', 'Treinador de Sala'),
  ('3', 'Treinador de Grupo'),
  ('3', 'Treinador Privado'),
  ('4', 'Treinador de Grupo'),
  ('5', 'Treinador de Sala'),
  ('5', 'Treinador de Grupo'),
  ('6', 'Treinador de Grupo'),
  ('6', 'Treinador Privado');

INSERT INTO Ginasio (Nome, DataAbertura, Rua, CodigoPostal, Cidade, Distrito)
VALUES
  ('Motivation Gym', '2023-01-01', 'Rua Augusta', '4700', 'Braga', 'Braga');
```

Figura 47 - Povoamento da tabela Ginasio

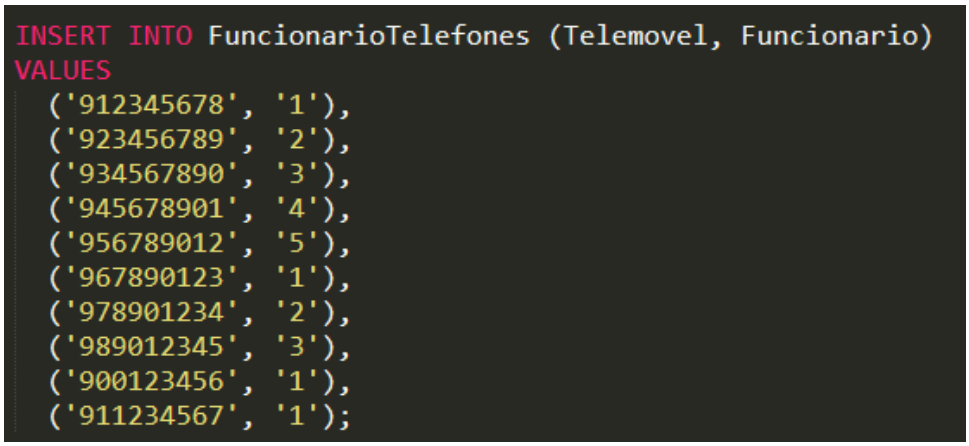
```sql
-- Inserir dados na tabela FuncionarioGinasio
INSERT INTO FuncionarioGinasio (DataPagamento, SalarioBase, SalarioTotal, HorarioTrabalho, Funcionario, Ginasio)
VALUES
  ('2022-01-01', '1500.0000', '1500.0000', '1', '1', '1'),
  ('2022-02-01', '1500.0000', '1500.0000', '1', '1', '1'),
  ('2022-02-01', '1200.0000', '1200.0000', '2', '2', '1'),
  ('2022-03-01', '1500.0000', '1500.0000', '1', '1', '1'),
  ('2022-03-01', '1200.0000', '1200.0000', '2', '2', '1'),
  ('2022-03-01', '900.0000', '900.0000', '3', '3', '1'),
  ('2022-04-01', '1500.0000', '1500.0000', '1', '1', '1'),
  ('2022-04-01', '1200.0000', '1200.0000', '2', '2', '1'),
  ('2022-04-01', '900.0000', '900.0000', '3', '3', '1'),
  ('2022-04-01', '1100.0000', '1100.0000', '1', '4', '1'),
  ('2022-05-01', '1500.0000', '1500.0000', '1', '1', '1'),
  ('2022-05-01', '1200.0000', '1200.0000', '2', '2', '1'),
  ('2022-05-01', '900.0000', '900.0000', '3', '3', '1'),
  ('2022-05-01', '1100.0000', '1100.0000', '1', '4', '1'),
  ('2022-05-01', '1000.0000', '1000.0000', '2', '5', '1'),
  ('2022-06-01', '1500.0000', '1500.0000', '1', '1', '1'),
  ('2022-06-01', '1200.0000', '1200.0000', '2', '2', '1'),
  ('2022-06-01', '900.0000', '900.0000', '3', '3', '1'),
  ('2022-06-01', '1100.0000', '1100.0000', '1', '4', '1'),
  ( '2022-06-01', '1000.0000', '1000.0000', '2', '5', '1'),
  ('2022-06-01', '1100.0000', '1100.0000', '1', '6', '1');
```

Figura 48 - Povoamento da tabela FuncionarioGinasio:



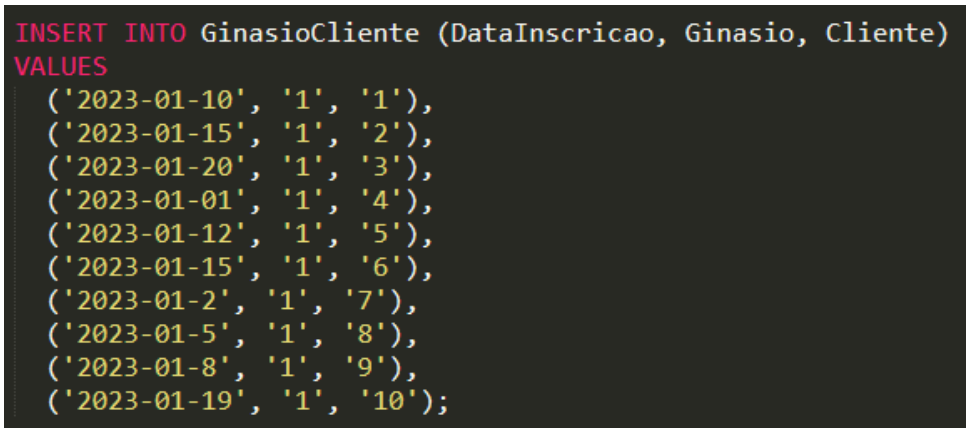
```
-- Inserir dados na tabela FuncionarioCliente
INSERT INTO FuncionarioCliente (HorarioInicio, HorarioFim, Espaco, Modalidade, Valor, Funcionario, Cliente, Ginasio)
VALUES
  ('2023-01-10 08:00:00', '2023-01-10 09:00:00', '1', 'Pilates', '2.50', '3', '1', '1'),
  ('2023-01-10 08:00:00', '2023-01-10 09:00:00', '1', 'Pilates', '2.50', '3', '2', '1'),
  ('2023-01-13 14:00:00', '2023-01-13 15:30:00', '3', 'Yoga', '12.75', '3', '3', '1'),
  ('2023-01-14 16:00:00', '2023-01-14 17:00:00', '4', 'Zumba', '8.50', '5', '4', '1'),
  ('2023-01-15 10:00:00', '2023-01-15 11:30:00', '5', 'Spinning', '9.75', '6', '5', '1'),
  ('2023-01-16 18:30:00', '2023-01-16 20:00:00', '2', 'CrossFit', '20.00', '6', '6', '1'),
  ('2023-01-17 07:30:00', '2023-01-17 08:30:00', '2', 'Personal Training', '15.50', '6', '7', '1'),
  ('2023-01-18 15:00:00', '2023-01-18 16:30:00', '3', 'Personal Training', '12.75', '3', '8', '1'),
  ('2023-01-19 17:30:00', '2023-01-19 18:30:00', '4', 'Zumba', '8.50', '5', '8', '1'),
  ('2023-01-20 09:00:00', '2023-01-20 10:30:00', '5', 'Spinning', '9.75', '6', '8', '1');
```

Figura 49 - Povoamento da tabela FuncionarioCliente:



```
INSERT INTO FuncionarioTelefones (Telemovel, Funcionario)
VALUES
  ('912345678', '1'),
  ('923456789', '2'),
  ('934567890', '3'),
  ('945678901', '4'),
  ('956789012', '5'),
  ('967890123', '1'),
  ('978901234', '2'),
  ('989012345', '3'),
  ('900123456', '1'),
  ('911234567', '1');
```

Figura 50 - Povoamento da tabela FuncionarioTelefones:



```
INSERT INTO GinasioCliente (DataInscricao, Ginasio, Cliente)
VALUES
  ('2023-01-10', '1', '1'),
  ('2023-01-15', '1', '2'),
  ('2023-01-20', '1', '3'),
  ('2023-01-01', '1', '4'),
  ('2023-01-12', '1', '5'),
  ('2023-01-15', '1', '6'),
  ('2023-01-2', '1', '7'),
  ('2023-01-5', '1', '8'),
  ('2023-01-8', '1', '9'),
  ('2023-01-19', '1', '10');
```

Figura 51 - Povoamento da tabela GinasioCliente:

Já relativamente ao povoamento automático, foi desenvolvido o seguinte código em phyton:

```python
import csv
import mysql.connector

# Function to read data from a CSV file and return as a list of dictionaries
def read_csv(filename):
    with open(filename, newline='') as file:
        reader = csv.DictReader(file)
        return list(reader)


# Function to execute SQL queries
def execute_query(connection, query):
    cursor = connection.cursor()
    cursor.execute(query)
    connection.commit()

# Establish database connection
connection = mysql.connector.connect(
    host='localhost',
    user='root',
    password='Animais1!',
    database='Ginasio'
)

# Read data from CSV files
plano_treino_data = read_csv('plano_treino.csv')
cliente_data = read_csv('cliente.csv')
cliente_telefones_data = read_csv('cliente_telefones.csv')
entradas_saidas_data = read_csv('entradas_saidas.csv')
fatura_data = read_csv('fatura.csv')
funcao_data = read_csv('funcao.csv')
funcionario_data = read_csv('funcionario.csv')
funcionario_funcao_data = read_csv('funcionario_funcao.csv')
ginasio_data = read_csv('ginasio.csv')
funcionario_ginasio_data = read_csv('funcionario_ginasio.csv')

# Populate tables

# PlanoTreino
for row in plano_treino_data:
    query = f"INSERT INTO PlanoTreino (Id, Preco, HorarioFim, HorarioInicio) VALUES ({row['Id']}, {row['Preco']}, '{row['HorarioFim']}', '{row['HorarioInicio']}')"
    execute_query(connection, query)

# Cliente
for row in cliente_data:
    query = f"INSERT INTO Cliente (NIF, DataNascimento, Email, Sexo, Nome, CodigoPostal, Rua, Cidade, Distrito, CondicoesMedicas, Extras, PlanoTreino) VALUES ('{row['NIF']}', '{row['DataNascimento']}', '{row['Email']}', '{row['Sexo']}', '{row['Nome']}', '{row['CodigoPostal']}', '{row['Rua']}', '{row['Cidade']}', '{row['Distrito']}', '{row['CondicoesMedicas']}', '{row['Extras']}', '{row['PlanoTreino']}')"
    execute_query(connection, query)

# ClienteTelefones
for row in cliente_telefones_data:
    query = f"INSERT INTO ClienteTelefones (Telemovel, Cliente) VALUES ('{row['Telemovel']}', '{row['Cliente']}')"
    execute_query(connection, query)

# EntradasSaidas
for row in entradas_saidas_data:
    query = f"INSERT INTO EntradasSaidas (Cliente, Data, Horario, EntradaSaida, Ginasio) VALUES ('{row['Cliente']}', '{row['Data']}', '{row['Horario']}', '{row['EntradaSaida']}', '{row['Ginasio']}')"
    execute_query(connection, query)

# Fatura
for row in fatura_data:
    query = f"INSERT INTO Fatura (MetodoPagamento, ValorMensalidade, DataEmissao, PrecoTotal, DataPagamento, Cliente) VALUES (NULL, '{row['ValorMensalidade']}', '{row['DataEmissao']}', '{row['PrecoTotal']}', NULL, '{row['Cliente']}')"
    execute_query(connection, query)

# Funcao
for row in funcao_data:
    query = f"INSERT INTO Funcao (Id, Nome) VALUES ({row['Id']}, '{row['Nome']}')"
    execute_query(connection, query)

# Funcionario
for row in funcionario_data:
    query = f"INSERT INTO Funcionario (NIF, DataNascimento, Email, Sexo, Nome, CodigoPostal, Rua, Cidade, Distrito, Salario, HorarioEntrada, HorarioSaida, Ginasio) VALUES ('{row['NIF']}', '{row['DataNascimento']}', '{row['Email']}', '{row['Sexo']}', '{row['Nome']}', '{row['CodigoPostal']}', '{row['Rua']}', '{row['Cidade']}', '{row['Distrito']}', '{row['Salario']}', '{row['HorarioEntrada']}', '{row['HorarioSaida']}', '{row['Ginasio']}')"
    execute_query(connection, query)

# FuncionarioFuncao
for row in funcionario_funcao_data:
    query = f"INSERT INTO FuncionarioFuncao (Funcionario, Funcao) VALUES ('{row['Funcionario']}', '{row['Funcao']}')"
    execute_query(connection, query)

# Ginasio
```

Figura 52 - Código em Python para mapeamento automático

Neste exemplo, o programa começa por se conectar à base de dados, depois cria um cursor, abre o ficheiro, faz parsing dos dados separados por vírgulas e por fim realizao povoamento de todas as tabelas após a execução do mesmo com “python povoamento.py”. O programa é ainda genérico, na medida que pode ser facilmente alterado para preencher outras tabelas.

7. Conclusão e Trabalho Futuro

O desenvolvimento de uma base de dados em SQL para um ginásio proporcionou uma solução eficiente para a gestão das informações essenciais ao funcionamento do estabelecimento. A base de dados fornece uma estrutura sólida e organizada para o armazenamento e manipulação dos dados.

Foi possível criar tabelas que representam as entidades-chave do ginásio, estabelecendo os relacionamentos necessários entre elas. Todas as queries implementadas possibilitam a obtenção de informações relevantes de forma ágil e eficiente. Dessa forma, é viável listar todos os clientes inscritos no ginásio, encontrar o instrutor responsável por uma aula específica e realizar diversos outros tipos de consultas personalizadas para atender às necessidades do ginásio.

Falamos com o Senhor Júnior e com a conclusão do projeto de base de dados a gestão do estabelecimento foi melhorada. Os processos de inscrição, pagamento, agendamento de aulas e acompanhamento da frequência dos clientes foram otimizados, resultando numa melhor experiencia para os clientes, como foi provado pelo último questionário realizado pela direção.

Em suma, o projeto de uma base de dados em SQL para um ginásio é uma solução indispensável para a gestão eficiente de informações relacionadas com os clientes, funcionários, aulas e equipamentos. Através da base de dados, é possível promover a segurança dos dados e otimizar o funcionamento do ginásio, garantindo a sua competividade e sucesso no mercado.

8. Referências Bibliográficas

Connolly, T., & Begg, C. (2015). Database Systems: A Practical Approach to Design, Implementation, and Management: Global Edition. Harlow, England: Pearson.