6.2 Criação Física da Base de Dados

```
CREATE SCHEMA IF NOT EXISTS `total_health` DEFAULT CHARACTER SET utf8 ;
USE `Total_health` ;


CREATE TABLE IF NOT EXISTS `Total_health`.`GrupoDeAlimentos` (
  `idGrupoAlimentos` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`idGrupoAlimentos`));


  CREATE TABLE IF NOT EXISTS `Total_health`.`Alimentos` (
  `idAlimentos` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(80) NOT NULL,
  `Quantidade` DOUBLE(3,1) NOT NULL,
  `GrupoDeAlimentos_idGrupoAlimentos` INT NOT NULL,
  PRIMARY KEY (`idAlimentos`),
    FOREIGN KEY (`GrupoDeAlimentos_idGrupoAlimentos`)
    REFERENCES `Total_health`.`GrupoDeAlimentos` (`idGrupoAlimentos`));


  CREATE TABLE IF NOT EXISTS `Total_health`.`Nutricionista` (
  `idNutricionista` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Nascimento` DATE NOT NULL,
```

```
`DataCadastro` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
`Especialidade` VARCHAR(30) NULL,
PRIMARY KEY (`idNutricionista`));


CREATE TABLE IF NOT EXISTS `Total_health`.`Usuario` (
`idUsuario` INT NOT NULL AUTO_INCREMENT,
`Nome` VARCHAR(45) NOT NULL,
`Endereco` VARCHAR(45) NOT NULL,
`Nascimento` DATE NULL,
`Sexo` CHAR(2) NOT NULL,
`Medicamentos` VARCHAR(320) NULL,
`Doencas` VARCHAR(320) NULL,
`Fuma` CHAR(2) NULL,
`Bebe` CHAR(2) NULL,
`IntoleranciaAlimentar` VARCHAR(320) NULL,
`Obs` VARCHAR(320) NULL,
`DataCadastro` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
`Funcao` VARCHAR(20) NULL,
`Nutricionista_idNutricionista` INT NOT NULL,
PRIMARY KEY (`idUsuario`),
  FOREIGN KEY (`Nutricionista_idNutricionista`)
  REFERENCES `Total_health`.`Nutricionista` (`idNutricionista`));
```

```
  CREATE TABLE IF NOT EXISTS `Total_health`.`NivelDeAcesso` (
`idNivelAcesso` INT NOT NULL AUTO_INCREMENT,
`NivelAcesso` VARCHAR(15) NULL,
PRIMARY KEY (`idNivelAcesso`));


CREATE TABLE IF NOT EXISTS `Total_health`.`Login` (
`idAcesso` INT NOT NULL AUTO_INCREMENT,
`Email` VARCHAR(30) NOT NULL,
`Senha` VARCHAR(50) NOT NULL,
`Nutricionista_idNutricionista` INT,
`Usuario_idUsuario` INT,
`NivelDeAcesso_idNivelAcesso` INT NOT NULL,
PRIMARY KEY (`idAcesso`),
  FOREIGN KEY (`Nutricionista_idNutricionista`)
  REFERENCES `Total_health`.`Nutricionista` (`idNutricionista`),
  FOREIGN KEY (`Usuario_idUsuario`)
  REFERENCES `Total_health`.`Usuario` (`idUsuario`),
  FOREIGN KEY (`NivelDeAcesso_idNivelAcesso`)
  REFERENCES `Total_health`.`NivelDeAcesso` (`idNivelAcesso`));
```

```
CREATE TABLE IF NOT EXISTS `Total_health`.`Telefones` (
  `idTelefones` INT NOT NULL AUTO_INCREMENT,
  `Fixo` INT NULL,
  `Celular` INT NULL,
  `Nutricionista_idNutricionista` INT,
  `Usuario_idUsuario` INT,
  PRIMARY KEY (`idTelefones`),
    FOREIGN KEY (`Nutricionista_idNutricionista`)
    REFERENCES `Total_health`.`Nutricionista` (`idNutricionista`),
    FOREIGN KEY (`Usuario_idUsuario`)
    REFERENCES `Total_health`.`Usuario` (`idUsuario`));


    CREATE TABLE IF NOT EXISTS `Total_health`.`CadastroDeDietas` (
  `idCadastroDeDietas` INT NOT NULL AUTO_INCREMENT,
  `Nutricionista_idNutricionista` INT NOT NULL,
  `Usuario_idUsuario` INT NOT NULL,
  PRIMARY KEY (`idCadastroDeDietas`),
    FOREIGN KEY (`Nutricionista_idNutricionista`)
    REFERENCES `Total_health`.`Nutricionista` (`idNutricionista`),
    FOREIGN KEY (`Usuario_idUsuario`)
    REFERENCES `Total_health`.`Usuario` (`idUsuario`));
```

```
CREATE TABLE IF NOT EXISTS `Total_health`.`CadastroDeDietas_tem_Alimentos` (
  `CadastroDeDietas_idCadastroDeDietas` INT NOT NULL,
  `Alimentos_idAlimentos` INT NOT NULL,
  `QuantidadeConsumida` DOUBLE(3,1) NULL,
  `ModoDePreparo` VARCHAR(15) NULL,
    FOREIGN KEY (`CadastroDeDietas_idCadastroDeDietas`)
    REFERENCES `Total_health`.`CadastroDeDietas` (`idCadastroDeDietas`),
    FOREIGN KEY (`Alimentos_idAlimentos`)
    REFERENCES `Total_health`.`Alimentos` (`idAlimentos`));


CREATE TABLE IF NOT EXISTS `Total_health`.`Recomendacoes` (
  `idRecomendacoes` INT NOT NULL AUTO_INCREMENT,
  `Descricao` VARCHAR(550) NULL,
  `Tipo` VARCHAR(15) NULL,
  `Consumo` double(3,2) NULL,
  `Data` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `Nutricionista_idNutricionista` INT NOT NULL,
  `Usuario_idUsuario` INT NOT NULL,
  PRIMARY KEY (`idRecomendacoes`),
    FOREIGN KEY (`Nutricionista_idNutricionista`)
    REFERENCES `Total_health`.`Nutricionista` (`idNutricionista`),
    FOREIGN KEY (`Usuario_idUsuario`)
    REFERENCES `Total_health`.`Usuario` (`idUsuario`));
```

```
CREATE TABLE IF NOT EXISTS `Data` (
  `idData` INT NOT NULL AUTO_INCREMENT,
  `data` DATE NOT NULL,
  `Recomendacoes_idRecomendacoes` INT NOT NULL,
  PRIMARY KEY (`idData`),
    FOREIGN KEY (`Recomendacoes_idRecomendacoes`)
    REFERENCES `Recomendacoes` (`idRecomendacoes`));


    CREATE TABLE IF NOT EXISTS `Total_health`.`Dieta_tem_datas` (
  `idDietas_tem_datas` INT NOT NULL AUTO_INCREMENT,
  `DataDieta` DATE NOT NULL,
  `CadastroDeDietas_idCadastroDeDietas` INT NOT NULL,
  PRIMARY KEY (`idDietas_tem_datas`),
    FOREIGN KEY (`CadastroDeDietas_idCadastroDeDietas`)
    REFERENCES `Total_health`.`CadastroDeDietas` (`idCadastroDeDietas`));
```

```
CREATE TABLE IF NOT EXISTS `Total_health`.`FeedBack` (
  `idFeedBack` INT NOT NULL AUTO_INCREMENT,
  `Feedback` VARCHAR(500) NOT NULL,
  `Data` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  `Usuario_idUsuario` INT NOT NULL,
  `Nutricionista_idNutricionista` INT NOT NULL,
  PRIMARY KEY (`idFeedBack`),
    FOREIGN KEY (`Usuario_idUsuario`)
    REFERENCES `Total_health`.`Usuario` (`idUsuario`),
    FOREIGN KEY (`Nutricionista_idNutricionista`)
    REFERENCES `Total_health`.`Nutricionista` (`idNutricionista`));
```

6.3 Dicionário de Dados

Alimentos

Comentários da tabela: Alimentos

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idAlimentos	int(11)	Não				
Nome	varchar(80)	Não				
Quantidade	double(3,1)	Não				
GrupoDeAlimentos_idGrupoAlimentos	int(11)	Não		GrupoDeAlimentos -> idGrupoAlimentos		

CadastroDeDietas

Comentários da tabela: CadastroDeDietas

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idCadastroDeDietas	int(11)	Não				
Nutricionista_idNutricionista	int(11)	Não		Nutricionista -> idNutricionista		
Usuario_idUsuario	int(11)	Não		Usuario -> idUsuario		

CadastroDeDietas_tem_Alimentos

Comentários da tabela: CadastroDeDietas_tem_Alimentos

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
CadastroDeDietas_idCadastroDeDietas	int(11)	Não		CadastroDeDietas -> idCadastroDeDietas		
Alimentos_idAlimentos	int(11)	Não		Alimentos -> idAlimentos		
QuantidadeConsumida	double(3,1)	Sim	*NULL*			
ModoDePreparo	varchar(15)	Sim	*NULL*			
Quantidade	double(3,1)	Sim	*NULL*			
Hora	varchar(8)	Não				

Data

Comentários da tabela: Data

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idData	int(11)	Não				
data	date	Não				
Recomendacoes_idRecomendacoes	int(11)	Não		Recomendacoes -> idRecomendacoes		

Dieta_tem_datas

Comentários da tabela: Dieta_tem_datas

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idDietas_tem_datas	int(11)	Não				
DataDieta	date	Não				
CadastroDeDietas_idCadastroDeDietas	int(11)	Não		CadastroDeDietas -> idCadastroDeDietas		

FeedBack

Comentários da tabela: FeedBack

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idFeedBack	int(11)	Não				
Feedback	varchar(500)	Não				
Data	timestamp	Sim	CURRENT_TIMESTAMP			
dataDieta	date	Não				
tipo	char(1)	Não				
Usuario_idUsuario	int(11)	Não		Usuario -> idUsuario		
Nutricionista_idNutricionista	int(11)	Não		Nutricionista -> idNutricionista		

GrupoDeAlimentos

Comentários da tabela: GrupoDeAlimentos

Coluna	Tipo	Nulo	Padrão	Comentários	MIME
idGrupoAlimentos	int(11)	Não			
Nome	varchar(30)	Não			

Login

Comentários da tabela: Login

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idAcesso	int(11)	Não				
Email	varchar(50)	Não				
Senha	varchar(50)	Não				
Nutricionista_idNutricionista	int(11)	Sim	*NULL*	Nutricionista -> idNutricionista		
Usuario_idUsuario	int(11)	Sim	*NULL*	Usuario -> idUsuario		
NivelDeAcesso_idNivelAcesso	int(11)	Não		NivelDeAcesso -> idNivelAcesso		

NivelDeAcesso

Comentários da tabela: NivelDeAcesso

Coluna	Tipo	Nulo	Padrão	Comentários	MIME
idNivelAcesso	int(11)	Não			
NivelAcesso	varchar(15)	Sim	*NULL*		

Nutricionista

Comentários da tabela: Nutricionista

Coluna	Tipo	Nulo	Padrão	Comentários	MIME
idNutricionista	int(11)	Não			
Nome	varchar(45)	Não			
Nascimento	date	Não			
DataCadastro	timestamp	Sim	CURRENT_TIMESTAMP		
Especialidade	varchar(34)	Sim	*NULL*		

Recomendacoes

Comentários da tabela: Recomendacoes

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idRecomendacoes	int(11)	Não				
Descricao	varchar(550)	Sim	*NULL*			
Tipo	varchar(15)	Sim	*NULL*			
Consumo	double(1,2)	Sim	*NULL*			
Data	timestamp	Não	CURRENT_TIMESTAMP			
Nutricionista_idNutricionista	int(11)	Não		Nutricionista -> idNutricionista		
Usuario_idUsuario	int(11)	Não		Usuario -> idUsuario		
Url	varchar(100)	Não				

Telefones

Comentários da tabela: Telefones

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idTelefones	int(11)	Não				
Fixo	varchar(15)	Sim	*NULL*			
Celular	varchar(15)	Sim	*NULL*			
Nutricionista_idNutricionista	int(11)	Sim	*NULL*	Nutricionista -> idNutricionista		
Usuario_idUsuario	int(11)	Sim	*NULL*	Usuario -> idUsuario		

Usuario

Comentários da tabela: Usuario

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idUsuario	int(11)	Não				
Nome	varchar(45)	Não				
Endereco	varchar(100)	Não				
Nascimento	date	Sim	*NULL*			
Sexo	char(2)	Não				
Medicamentos	varchar(320)	Sim	*NULL*			
Doencas	varchar(320)	Sim	*NULL*			
Fuma	varchar(5)	Sim	*NULL*			
Bebe	varchar(5)	Sim	*NULL*			
IntoleranciaAlimentar	varchar(320)	Sim	*NULL*			
Obs	varchar(320)	Sim	*NULL*			
DataCadastro	timestamp	Sim	CURRENT_TIMESTAMP			
Funcao	varchar(40)	Sim	*NULL*			
Nutricionista_idNutricionista	int(11)	Não		Nutricionista -> idNutricionista		

6.3.1 Store procedures

```
delimiter $$
create procedure cadastrarNutri (in nome_ varchar(45),
                                 in nascimento_ date,
                                 in especialidade_ varchar(34),
                                 in email_ varchar(30),
                                 in senha_ varchar(50),
                                 in tel_ varchar(15))
begin

  if( select count(*) from Login where Email = email_ and NivelDeAcesso_idNivelAcesso = 2 > 0 ) then
      signal sqlstate '45000'
    set message_text = 'existe';
  else
  insert into Nutricionista (Nome,Nascimento,DataCadastro,Especialidade)
         values (nome_,nascimento_,current_timestamp - interval 3 hour,especialidade_);


    select max(idNutricionista) into @idNutri_ from Nutricionista;

    insert into Login (Email,Senha,Nutricionista_idNutricionista,NivelDeAcesso_idNivelAcesso)
         values (email_,senha_,@idNutri_,2);

  insert into Telefones (Celular,Nutricionista_idNutricionista) values (tel_,@idNutri_);
    end if;

end $$
delimiter ;
```

Figura 23- Procedure para cadastrar nutricionista

```
delimiter $$
create procedure cadastrarUsuarios (in nome_ varchar(45),
                                    in endereco_ varchar(45),
                                    in nasc_ date,
                                    in sexo_ char(2),
                                    in medicamentos_ varchar(320),
                                    in doencas_ varchar(320),
                                    in fuma_ varchar(5),
                                    in bebe_ varchar(5),
                                    in IntoleranciaAlimentar_ varchar(320),
                                    in obs_ varchar(320),
                                    in funcao_ varchar(20),
                                    in nutriID_ int,
                                    in email_ varchar(30),
                                    in senha_ varchar(50),
                                    in telFixo_ varchar(15),
                                    in telCel_ varchar(15))
begin

  insert into Usuario (Nome,Endereco,Nascimento,Sexo,Medicamentos,Doencas,Fuma,Bebe,IntoleranciaAlimentar,Obs,DataCadastro,Funcao,Nutricionista_idNutricionista)
         values (nome_,endereco_,nasc_,sexo_,medicamentos_,doencas_,fuma_,bebe_,IntoleranciaAlimentar_,obs_,current_timestamp - interval 3 hour,funcao_,nutriID_);


    select max(idUsuario) into @idUser_ from Usuario;

    insert into Login (Email,Senha,Usuario_idUsuario,NivelDeAcesso_idNivelAcesso)
         values (email_,senha_,@idUser_,3);

  insert into Telefones (Fixo,Celular,Usuario_idUsuario) values (telFixo_,telCel_,@idUser_);


end $$
delimiter ;
```

Figura 24- Procedure para cadastrar usuários

```
delimiter $$
create procedure CadastrarConsumo(in quantiConsumo_ double,
                                  in modoPreparo_ varchar(15),
                                  in idDietas_ int,
                                  in idAlimento_ int,
                                  in hora_ varchar(9))
  begin

    SELECT CURTIME() - interval 3 hour into @hora_;

    if(SELECT COUNT( * ) FROM `CadastroDeDietas_tem_Alimentos` WHERE QuantidadeConsumida != 'null'
      and Alimentos_idAlimentos = idAlimento_ and  CadastroDeDietas_idCadastroDeDietas = idDietas_ > 0)then

            select distinct Quantidade into @quantidade_ from CadastroDeDietas_tem_Alimentos
            where Alimentos_idAlimentos = idAlimento_
            and CadastroDeDietas_idCadastroDeDietas = idDietas_;


              insert into CadastroDeDietas_tem_Alimentos (CadastroDeDietas_idCadastroDeDietas, Alimentos_idAlimentos, QuantidadeConsumida,  ModoDePreparo, Quantidade, Hora)
                values(idDietas_, idAlimento_, quantiConsumo_, modoPreparo_, @quantidade_, hora_);
    else
            update CadastroDeDietas_tem_Alimentos set QuantidadeConsumida = quantiConsumo_, ModoDePreparo = modoPreparo_, Hora = hora_
            where CadastroDeDietas_idCadastroDeDietas = idDietas_
            and Alimentos_idAlimentos = idAlimento_;
    end if;
  end $$
delimiter ;
```

Figura 25- Procedure para cadastrar consumo

```
delimiter $$
create procedure cadastrarRecomendacoes(in descricao_ varchar(550),
                                        in tipo_ varchar(15),
                                        in nutricionistaID_ int,
                                        in usuarioID_ int,
                                        in data_ date,
                                        in url_ varchar(100))

begin
select max(idRecomendacoes) into @ultimo from Recomendacoes;

    if((SELECT COUNT(*)
          FROM  `Recomendacoes`
          WHERE Descricao = descricao_ ) = 0) then


        INSERT INTO Recomendacoes (Descricao, Tipo, Consumo, Data, Nutricionista_idNutricionista, Usuario_idUsuario, Url)
        VALUES (descricao_, tipo_, 0, current_timestamp - interval 3 hour, nutricionistaID_, usuarioID_, url_);


        SELECT MAX(idRecomendacoes) INTO @idRecomendacoes from Recomendacoes;

        INSERT INTO Data (data,Recomendacoes_idRecomendacoes) values (data_, @idRecomendacoes);

else

  INSERT INTO Data (data,Recomendacoes_idRecomendacoes) values (data_, @ultimo);

end if;


end $$

delimiter ;
```

Figura 26 - Procedure para cadastrar recomendações

```
delimiter $$
create procedure cadastrarConsumoRecomendacoes(in Consumo_ double(3,2),
                                               in usuarioID_ int,
                                               in Tipo_ varchar(15))
begin

    UPDATE Recomendacoes
    SET Consumo = Consumo + Consumo_
    WHERE Usuario_idUsuario = usuarioID_
        and Tipo = Tipo_;

end $$
delimiter ;
```

Figura 27- Procedure cadastrar consumo de recomendações

```
delimiter $$
create procedure cadastrarFeedBack(in feedback_ varchar(500),
                                   in usuarioID_ int,
                                   in nutricionistaID_ int,
                                   in tipo_ char(1),
                                   in dataDieta_ date)
begin


   INSERT INTO FeedBack (Feedback, Data, dataDieta, tipo, Usuario_idUsuario, Nutricionista_idNutricionista)
   VALUES (feedback_,current_timestamp() - interval 3 hour, dataDieta_, tipo_, usuarioID_, nutricionistaID_);


end $$
delimiter ;
```

Figura 28- Procedure para cadastrar FeedBack

```
delimiter $$
create procedure recuperarSenha(in email_ varchar(30),
                                in senha_ varchar(50))
begin

  if((select count(*) from Login where Email = email_) > 0)then
    UPDATE Login SET Senha = senha_
        where Email = email_;
  end if;
end $$
delimiter ;
```

Figura 29- Procedure para recuperar senha