BASE DE DADOS DA ESCOLA DE CONDUÇÃO "DRIVE IT".

GRUPO 18:
GONÇALO NOGUEIRA,A86617
JOANA SOUSA,A83614
JOÃO FREITAS,A3782
TIAGO GOMES,A78141

GRUPO 18

GONÇALO NOGUEIRA

a86617

JOÃO FREITAS

a83782

JOANA SOUSA

a83614

TIAGO GOMES

a78141

03

BASE DE DADOS

ORLANDO BELO

ANDRÉ FERREIRA

HUGO GUIMARÃES

JOÃO COELHO

DOCENTES

ÍNDICE

O PORQUÊ DA ESCOLHA

- A escola de condução Drive It, surgiu com a enorme vontade do Sr. Gonçalves de ter a sua própria escola de condução, ainda por cima sendo a primeira na cidade.

- Começou a sua atividade há 1 ano e verificou-se um aumento elevado devido ao número de clientes satisfeitos pela simpatia.

FUNDAMENTAÇÃO PARA A IMPLEMENTAÇÃO DA BASE DE DADOS

- Aumento de clientes.
- Armazenamento de uma maior quantidade de dados.
- Melhor funcionamento das aulas.
- Melhor gestão de instrutores, carros e clientes.
- Inviabilidade de gestão a nível manual.

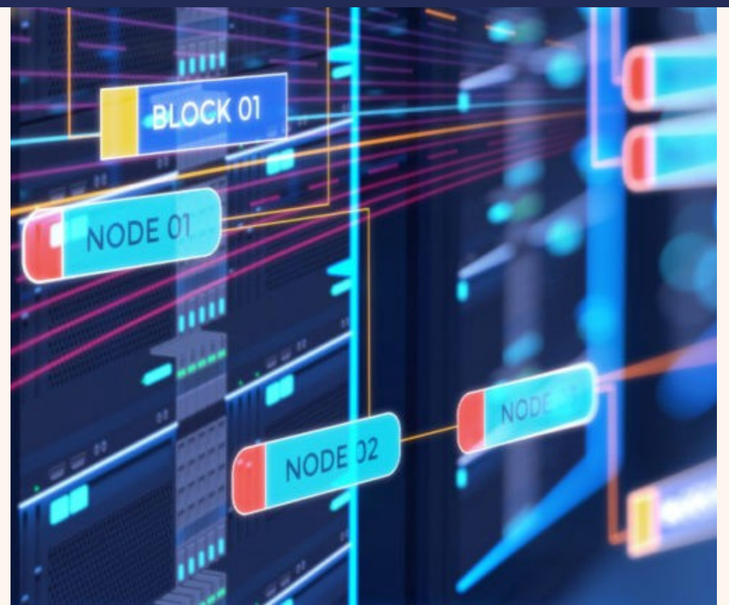
ANÁLISE DA VIABILIDADE DO PROCESSO

- Organizar de melhor forma a marcação de aulas de cada aluno, tal como cada instrutor associado a um carro e por sua vez a um cliente.

- Tornar o negócio mais eficiente e organizado deixando desta forma o processo mais rápido e cómodo.

MÉTODO DE LEVANTAMENTO E ANÁLISE DE REQUISITOS ADOTADOS

- Várias entrevistas para se perceber necessidades. Retiraram-se entidades requeridas.
- Forma como se interligam os tipos de dados.
- Agrupar todos os requisitos correspondentes a cada perfil de utilização.



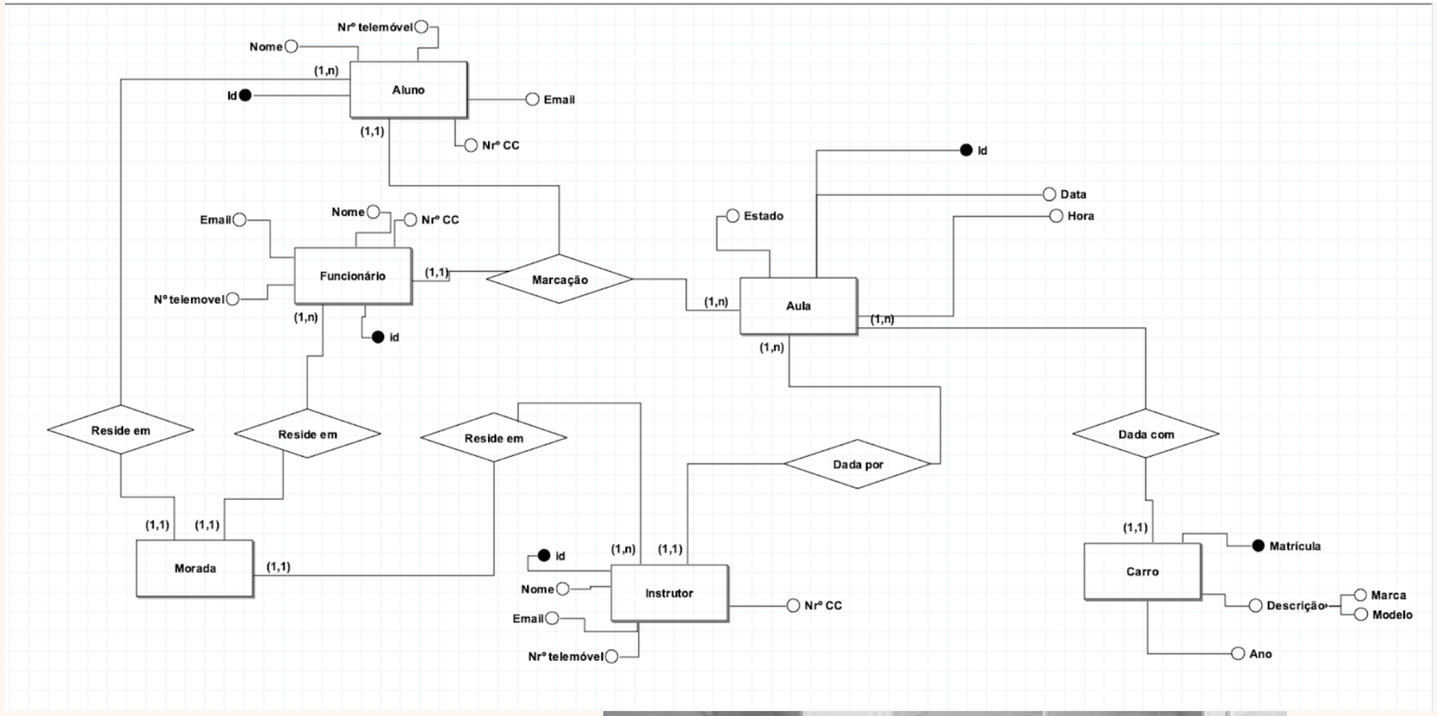
MÉTODO DE LEVANTAMENTO E ANÁLISE DE REQUISITOS ADOTADOS

- Informações(Funcionários, Instrutores, Alunos, Carros e Aulas).
- Marcação de Aulas.
- Horário de marcações.
- Consultas(Aulas, Carros).

MODELO CONCEPTUAL

- Entidades
- Relacionamentos
- Atributos

**Estado* - L(Livre)/R(Realizada)/M(Marcada)/D(A Decorrer)

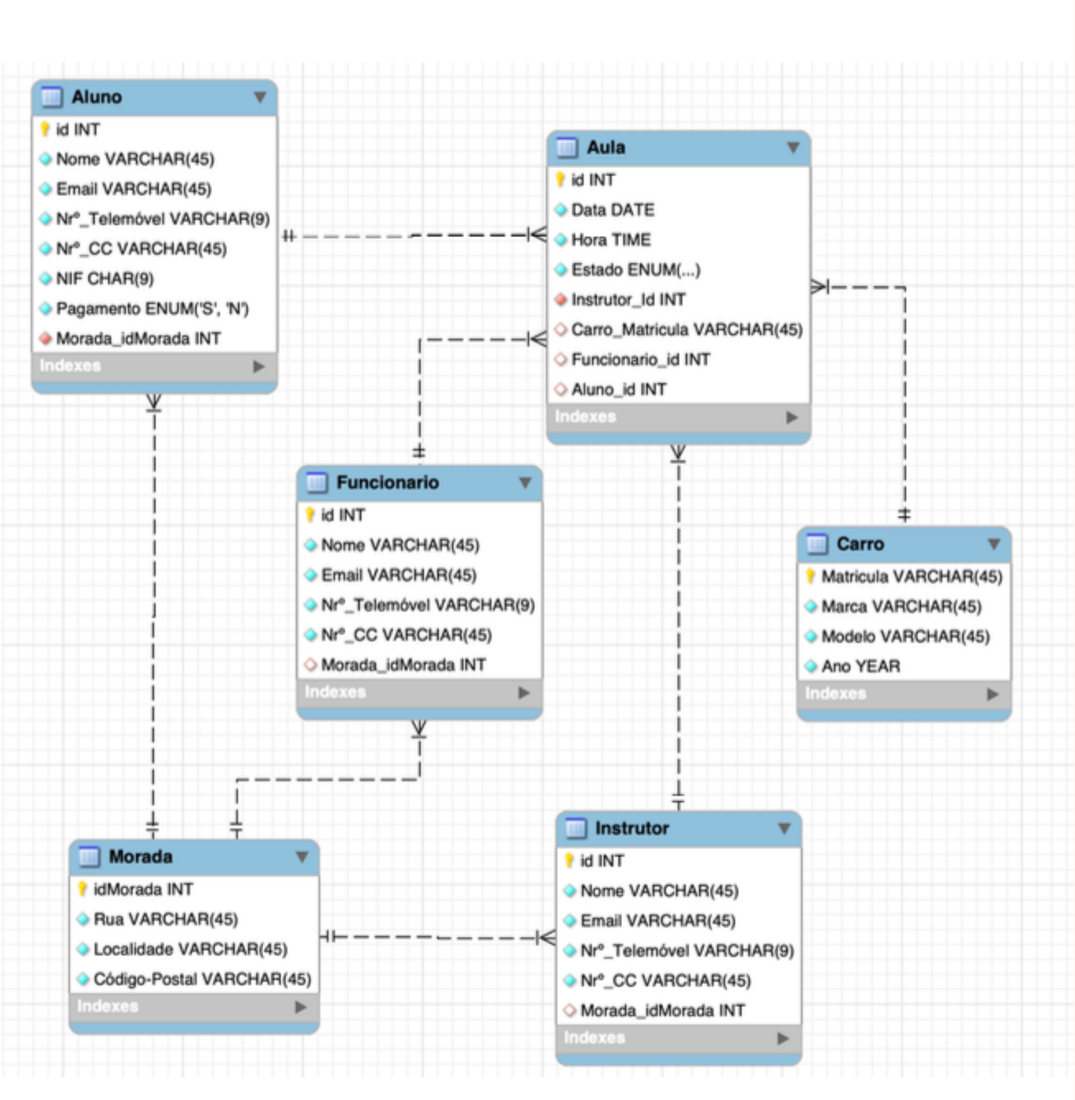


MODELO LÓGICO

- 1:N-Chave Primária na Entidade "pai" é utilizada como chave estrangeira na entidade "filho".

- N:M- Este tipo de relacionamento foi descartado durante a fase de conversão uma vez que origina uma nova tabela e não existe este relacionamento no modelo conceptual.

- A relação ternária "Marcação" foi incluída na relação aula (contém o id do aluno que reservou a aula e o id do funcionário que tratou a marcação). Sendo assim a relação Aula contém toda a informação necessária sobre as aulas.



INTEGRIDADE DA ENTIDADE

Nenhuma chave primária é nula.

INTEGRIDADE REFERENCIAL

A chave estrangeira possui um valor, esse valor deve referir-se a um tuplo
existente na relação pai. Exemplo:
"Instrutor_id" e "Aula_Instrutor".

DADOS REQUERIDOS

Alguns atributos não permitem valores nulos

DOMÍNIO DE ATRIBUTOS

Cada atributo pode ter apenas valores definidos num dado conjunto

RESTRIÇÕES ÀS ENTIDADES

Um Aluno só pode ter 3 Aulas por Dia

RESTRIÇÕES DE INTEGRIDADE

```sql
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema driveIt
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema driveIt
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `driveIt` ;
CREATE SCHEMA IF NOT EXISTS `driveIt` DEFAULT CHARACTER SET utf8 ;
USE `driveIt` ;

-- -----------------------------------------------------
-- Table `driveIt`.`Morada`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `driveIt`.`Morada` ;
CREATE TABLE IF NOT EXISTS `driveIt`.`Morada` (
  `idMorada` INT NOT NULL AUTO_INCREMENT,
  `Rua` VARCHAR(45) NOT NULL,
  `Localidade` VARCHAR(45) NOT NULL,
  `Código-Postal` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idMorada`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `driveIt`.`Funcionario`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `driveIt`.`Funcionario` ;
CREATE TABLE IF NOT EXISTS `driveIt`.`Funcionario` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `Nr°_Telemóvel` VARCHAR(9) NOT NULL,
  `Nr°_CC` VARCHAR(45) NOT NULL,
  `Morada_idMorada` INT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Funcionario_Morada1_idx` (`Morada_idMorada` ASC) VISIBLE,
  CONSTRAINT `fk_Funcionario_Morada1`
    FOREIGN KEY (`Morada_idMorada`)
    REFERENCES `driveIt`.`Morada` (`idMorada`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `driveIt`.`Aluno`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `driveIt`.`Aluno` ;
CREATE TABLE IF NOT EXISTS `driveIt`.`Aluno` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `Nr°_Telemóvel` VARCHAR(9) NOT NULL,
  `Nr°_CC` VARCHAR(45) NOT NULL,
  `NIF` CHAR(9) NOT NULL,
  `Pagamento` ENUM('S', 'N') NOT NULL,
  `Morada_idMorada` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Aluno_Morada1_idx` (`Morada_idMorada` ASC) VISIBLE,
  CONSTRAINT `fk_Aluno_Morada1`
    FOREIGN KEY (`Morada_idMorada`)
    REFERENCES `driveIt`.`Morada` (`idMorada`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `driveIt`.`Instrutor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `driveIt`.`Instrutor` ;
CREATE TABLE IF NOT EXISTS `driveIt`.`Instrutor` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `Nr°_Telemóvel` VARCHAR(9) NOT NULL,
  `Nr°_CC` VARCHAR(45) NOT NULL,
  `Morada_idMorada` INT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Instrutor_Morada1_idx` (`Morada_idMorada` ASC) VISIBLE,
  CONSTRAINT `fk_Instrutor_Morada1`
    FOREIGN KEY (`Morada_idMorada`)
    REFERENCES `driveIt`.`Morada` (`idMorada`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `driveIt`.`Carro`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `driveIt`.`Carro` ;
CREATE TABLE IF NOT EXISTS `driveIt`.`Carro` (
  `Matricula` VARCHAR(45) NOT NULL,
  `Marca` VARCHAR(45) NOT NULL,
  `Modelo` VARCHAR(45) NOT NULL,
  `Ano` YEAR(4) NOT NULL,
  PRIMARY KEY (`Matricula`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `driveIt`.`Aula`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `driveIt`.`Aula` ;
CREATE TABLE IF NOT EXISTS `driveIt`.`Aula` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Data` DATE NOT NULL,
  `Hora` TIME NOT NULL,
  -- L : livre; R: realizada; M: marcada; D : decorrer
  `Estado` ENUM('L', 'R', 'M', 'D') NOT NULL,
  `Instrutor_Id` INT DEFAULT 0 NOT NULL,
  `Carro_Matricula` VARCHAR(45) NULL,
  `Funcionario_id` INT NULL,
  `Aluno_id` INT NULL,
  PRIMARY KEY (`id`),
  INDEX `Instrutor_Id_idx` (`Instrutor_Id` ASC) VISIBLE,
  INDEX `fk_Aula_Carro1_idx` (`Carro_Matricula` ASC) VISIBLE,
  INDEX `fk_Aula_Funcionario1_idx` (`Funcionario_id` ASC) VISIBLE,
  INDEX `fk_Aula_Aluno1_idx` (`Aluno_id` ASC) VISIBLE,
  CONSTRAINT `Instrutor_Id`
    FOREIGN KEY (`Instrutor_Id`)
    REFERENCES `driveIt`.`Instrutor` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Aula_Carro1`
    FOREIGN KEY (`Carro_Matricula`)
    REFERENCES `driveIt`.`Carro` (`Matricula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Aula_Funcionario1`
    FOREIGN KEY (`Funcionario_id`)
    REFERENCES `driveIt`.`Funcionario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Aula_Aluno1`
    FOREIGN KEY (`Aluno_id`)
    REFERENCES `driveIt`.`Aluno` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

MODELO FÍSICO

Criação de Tabelas no MySQLWorkbench, que seguidamente,utilizando a ferramenta de Forward Engineering, permitiu a construção de componentes de baixo nível, com base numa abstração de alto nível.Seguidamente foi apenas necessário executar o código obtido.

GASTO DE MEMÓRIA

Nas condições indicadas anteriormente, a base de dados diariamente aumenta o gasto de memória em:

162*5(5 novos alunos)+ 70*25(espaços de aulas)= 2560 bytes (média diária)

De acordo com a povoação realizada pelo grupo a base de dados ocupa:

Total estimado de memória = 152*5(instrutores) + 152*5(funcionários) + 162*40 (alunos) + 139*50(moradas), 139*10(carros) + 70*175(aulas) = 28590 bytes =28.590 Kb=0.028590 Mb.

QUERIES

```sql
-- Querie 0: Lista de todas as aulas
Select * From aula;
```

```sql
-- Querie 1: Lista de slots das aulas livres com info relevante, sem identificar o nome do instrutor
Select Data, Hora, Estado, Instrutor_id AS `ID do Instrutor` From aula
    where Estado IN ('L');
```

```sql
-- Querie 2: Lista de slots das aulas livres com info relevante, indentificando o nome do instrutor
Select Instrutor_id AS `ID do Instrutor`, Instrutor.Nome AS `Nome do Instrutor`, Data, Hora, Estado  From aula
    JOIN Instrutor ON Instrutor.id = aula.Instrutor_id
        where Estado IN ('L');
```

```sql
-- Querie 3: Lista de aulas marcadas por Aluno, ordenados ascendentemente pelo nome do aluno (mostra morada também)
Select aula.id AS `ID da Aula`, Aluno.id AS `ID do Aluno`, Aluno.Nome, aula.Data, aula.Hora,
    aula.Carro_Matricula AS `Matrícula do Carro`, aula.Instrutor_id AS `ID do Instrutor`, Morada.Rua, Morada.Localidade,
    Morada.`Código-Postal` FROM Aluno
    JOIN aula ON Aluno.id = aula.Aluno_id
    JOIN Morada ON Morada.idMorada = Aluno.Morada_idMorada
        order by Aluno.Nome ASC;
```

```sql
-- Querie 7: Ranking de funcionário, consoante as aulas com que interagiu (exclui-se as que estão livres) por ordem
-- decrescente
Select Nome, Email, Nrº_Telemóvel AS `Nº de Telemóvel`, count(Funcionario.id) AS `Total de Aulas` From Funcionario
    JOIN aula ON aula.Funcionario_id = Funcionario.id
        where aula.Estado IN ('M','D','R')
        group by Funcionario.id
        order by `Total de Aulas` DESC;
```

PROCEDURES

```
-- Querie 16: Exemplo de uma marcação de aula, passar para 'marcada'. Só passa para marcada se o estado
-- for a livre e os dados introduzidos derem match
-- Se for preciso apagar a procedure:
DROP PROCEDURE marcar_aula;
delimiter //
CREATE PROCEDURE marcar_aula (IN idAula INT, idFuncionario INT, idAluno INT, matricula VARCHAR(45))
      BEGIN
       Update aula
           Set Estado = 'M', `Carro_Matricula` = matricula, Funcionario_Id = idFuncionario, Aluno_id = idAluno
           -- é dado o id do aluno, o dia, hora da aula e confirmar que se tratava de uma aula a decorrer
           where Estado = 'L' AND aula.id = idAula;
      END//
delimiter ;
```

```
-- Querie 15: Exemplo de um cancelamento de uma aula, passar para 'livre'. Só passa para livre se o estado
-- for de marcada e os dados introduzidos derem match
-- Se for preciso apagar a procedure:
DROP PROCEDURE cancelar_aula;
delimiter //
CREATE PROCEDURE cancelar_aula (IN idFuncionario INT, idAula INT)
      BEGIN
       Update aula
           -- atualiza slot para Livre
           -- atualiza a matricula do carro para null
           -- apaga o funcionário
           -- apaga o id do aluno
           Set Estado = 'L', Carro_Matricula = null, Funcionario_id = null, Aluno_id = null
           -- Aluno_id = 0,
               -- é dado o estado da aula, id do Funcionario (só funcionários é que cancelam aulas) e o id da aula
               where aula.Estado = 'M' AND aula.id = idAula AND aula.Funcionario_id = idFuncionario;
      END//
delimiter ;
```

```
-- Querie 17: Marcar slots de todas as aulas de um instrutor para um dia
DROP PROCEDURE marcar_slots_aula;
delimiter //
CREATE PROCEDURE marcar_slots_aula (IN idInstrutor INT, dataC DATE)
      BEGIN
       INSERT INTO aula (`Data`,`Hora`,`Estado`,`Instrutor_Id`)
           VALUES
               (dataC,"10:00:00","L",idInstrutor),
               (dataC,"11:00:00","L",idInstrutor),
               (dataC,"15:00:00","L",idInstrutor),
               (dataC,"16:00:00","L",idInstrutor),
               (dataC,"17:00:00","L",idInstrutor);
      END//
delimiter ;
```

PROCEDURE E VIEW

```
-- Querie 18: Verificar todas as aula no dia atual (utilizando views)
CREATE VIEW vwAulasHoje AS
Select * from aula
    where Data = date(now());

Select * from vwAulasHoje;
```

```
-- Querie 14: Exemplo de uma atualização de uma aula, passar para 'realizada'. Só passa para realizada se o estado
-- for a decorrer e os dados introduzidos derem match
-- Se for preciso apagar a procedure:
DROP PROCEDURE update_aula_realizada;
delimiter //
CREATE PROCEDURE update_aula_realizada (IN idInstrutor INT, idAula INT)
       BEGIN
        Update aula
            Set Estado = 'R'
            -- é dado o estado da aula, id do Instrutor, o dia e a hora da aula
            where Estado = 'D' AND aula.id = idAula;
       END//
delimiter ;
```

CONCLUSÃO

Desafiante