Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática

DigiPeet

Base De Dados

Março
2021

Conteúdo

1 Introdução:

Este relatório tem como objetivo demonstrar como foi implementada a base de dados para aplicação DigiPeet. A aplicação é uma ferramenta para o utilizador, que tem como objetivo facilitar a sua experiência. Nela o ultizador poderá criar uma conta pessoal, fazer gestão do armazém/stock, aceder a um calendário de atividades e configurar o Feed eet.

2 Descrição Geral:

A base de dados tem com função guardar e atualizar autonomamente todos os dados provenientes da aplicação e do dispensador, garantindo assim o seu bom funcionamento.

3 Tipo de Utilizadores

3.1 Administrador:

- Tem a função criar ou editar os utilizadores da aplicação;

3.2 Voluntário:

- Utilizadores da aplicação.

4 Descrição dos Use-Cases:

4.1 Todos os Utilizadores:

- Login: O utilizador entra na aplicação utilizando o email e a password escolhidos durante o processo de registo, o sistema regista a hora de entrada e apresenta ao utilizador a sua área de trabalho (Cada tipo de utilizador tem um login diferente).

- Logout: O utilizador sai da aplicação, o sistema regista a hora de saída, e é fechada a área de trabalho.

4.2 Administrador:

- Aprovar Contas: Aprovar o registo de novas contas na aplicação. Só depois deste passo é que as contas são adicionadas a base de dados.

- Editar Contas: Permite activar/desactivar as contas existentes, bem como alterar o nome e a password

- Lista das Contas: Obter uma listagem com todas as contas de utilizadores.

4.3 Voluntário:

4.3.1 Registo na app:

- Criar Conta: Preenchimento de um formulário com nome completo, email, idade, género, morada, telefone e tipo de utilizador.

-Validação da conta: É necessário validar o registo através do emai

4.3.2 Menu:

-Ler código de barras: O volutário pode ler códigos de barras de produtos através da camera do smartphone, para isso o utilizador precisa aceitar as permissões de acesso à camera.

-Adicionar produtos: Após a leitura do código de barras, caso esse produto não exista na aplicação, o utilizador pode adicioná-lo.

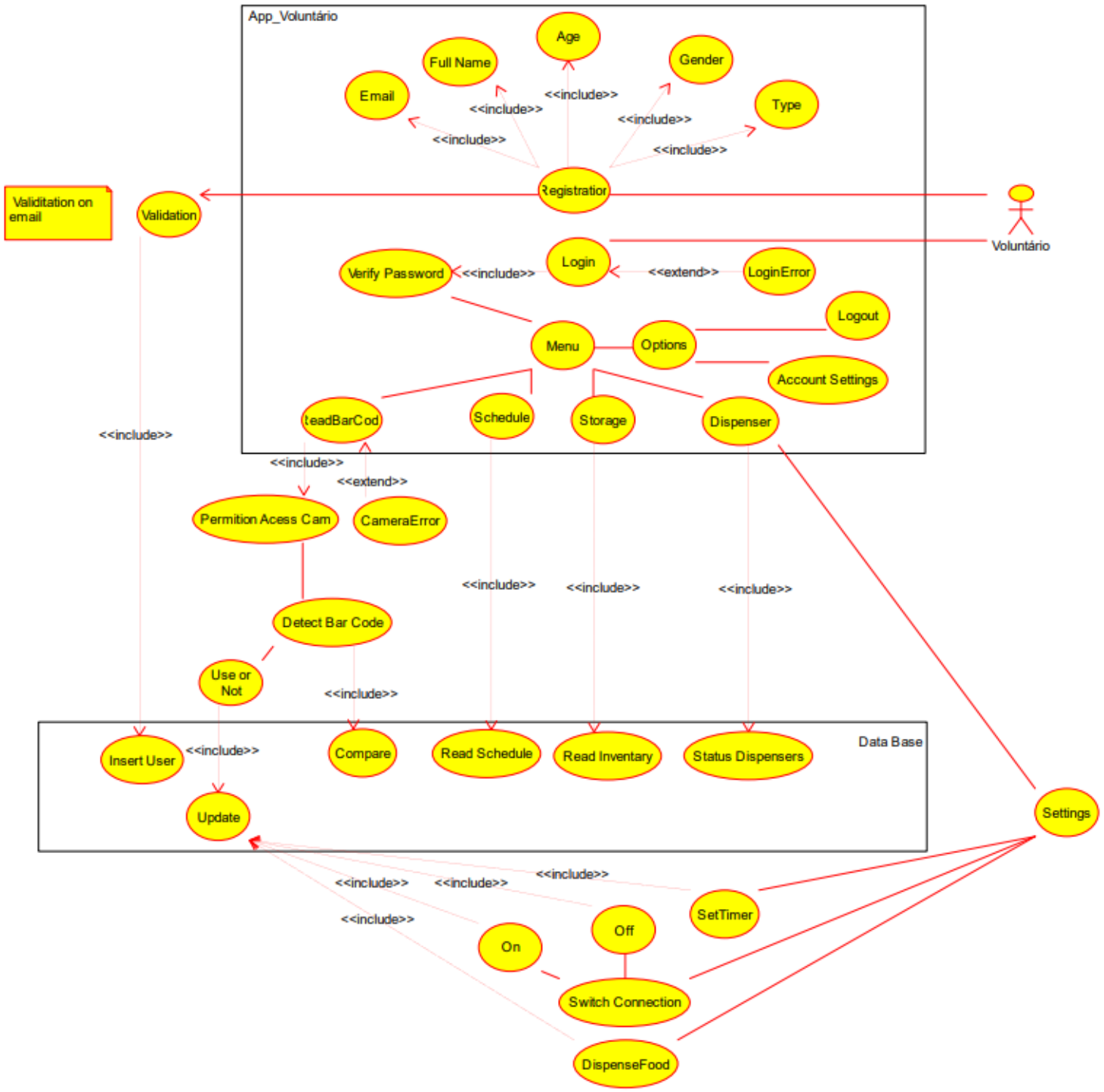
-Aceder ao calendário: O utilizador pode aceder ao calendário e ver as ativades futuras.

-Ver inventário: O utilizador pode ver os produtos que existem em armazém.

-Atualizar stock: O utilizador pode adicionar/retirar produtos ao stock.

-Aceder ao dispensador: O utilizador pode controlar o dispensador a partir da aplicação.

-Alterar definições: O utilizador pode alterar as definições da aplicação.


App_Voluntário
Age
Full Name
Gender
Email
Type
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
Registration
Validitation on email
Validation
Voluntário
Verify Password
<<include>>
Login
<<extend>>
LoginError
Logout
Menu
Options
Account Settings
teadBarCod
Schedule
Storage
Dispenser
<<include>>
<<include>>
<<extend>>
Permition Acess Cam
CameraError
<<include>>
<<include>>
<<include>>
Detect Bar Code
Use or Not
<<include>>
Data Base
Insert User
<<include>>
Compare
Read Schedule
Read Inventary
Status Dispensers
Update
Settings
<<include>>
<<include>>
<<include>>
SetTimer
<<include>>
On
Off
Switch Connection
DispenseFood

5 Descrição do Modelo de entidades:

O domínio da base de dados está representado no seguinte modelo de entidades.

No diagrama apresentamos uma classe **"Utilizador"**, que representa o todos os utilizadores, sendo que nela são registadas todas as informações do mesmo. Estas informações estão implicitas nos atributos: **name**, **email**, **password**, **age**, **gender**, **address**, **phone**, **type_user status**.

A cada utilizador está associado um login. Na classe **"Login"**
é registada a hora em que o utilizador entra no sistema e ao abandonar, através de logoff, é registada também a hora de saída. Estas informações estão implicitas nos atributos: **time_in_c** e **time_out_c**.

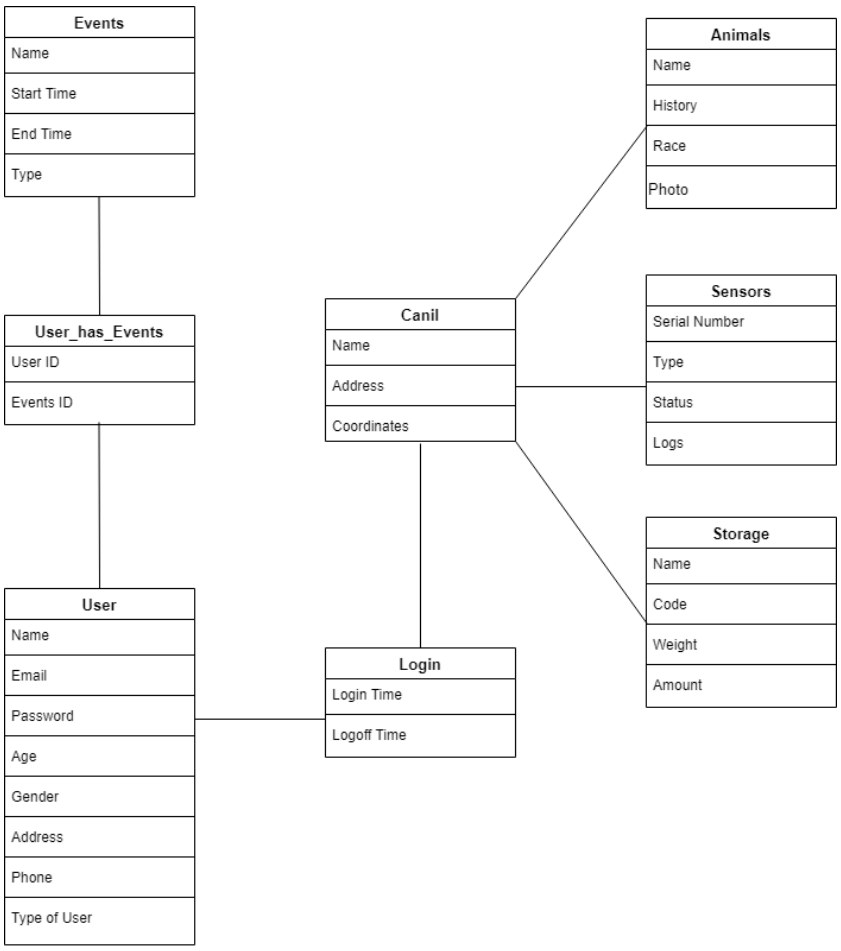
O utilizador pode aceder ao seu próprio calendário. Na classe **"Events"** são apresentadas algumas informações tais como o nome do evento, o seu tipo e a sua hora de início e de fim. Estas informações estão implicitas nos atributos: **event_init**, **event_fin**, **event_type** e **event_name**.

A classe **"Cannil"** está associada, tal como o nome indica, a todas as informações relacionadas com os canis, tais como, nome do canil, morada e coordenadas. Estas informações estão implicitas nos atributos: **name_cannil**, **adress_cannil** e **coordinates**. Cada canil tem os seus próprios sensores, animais e armazém.

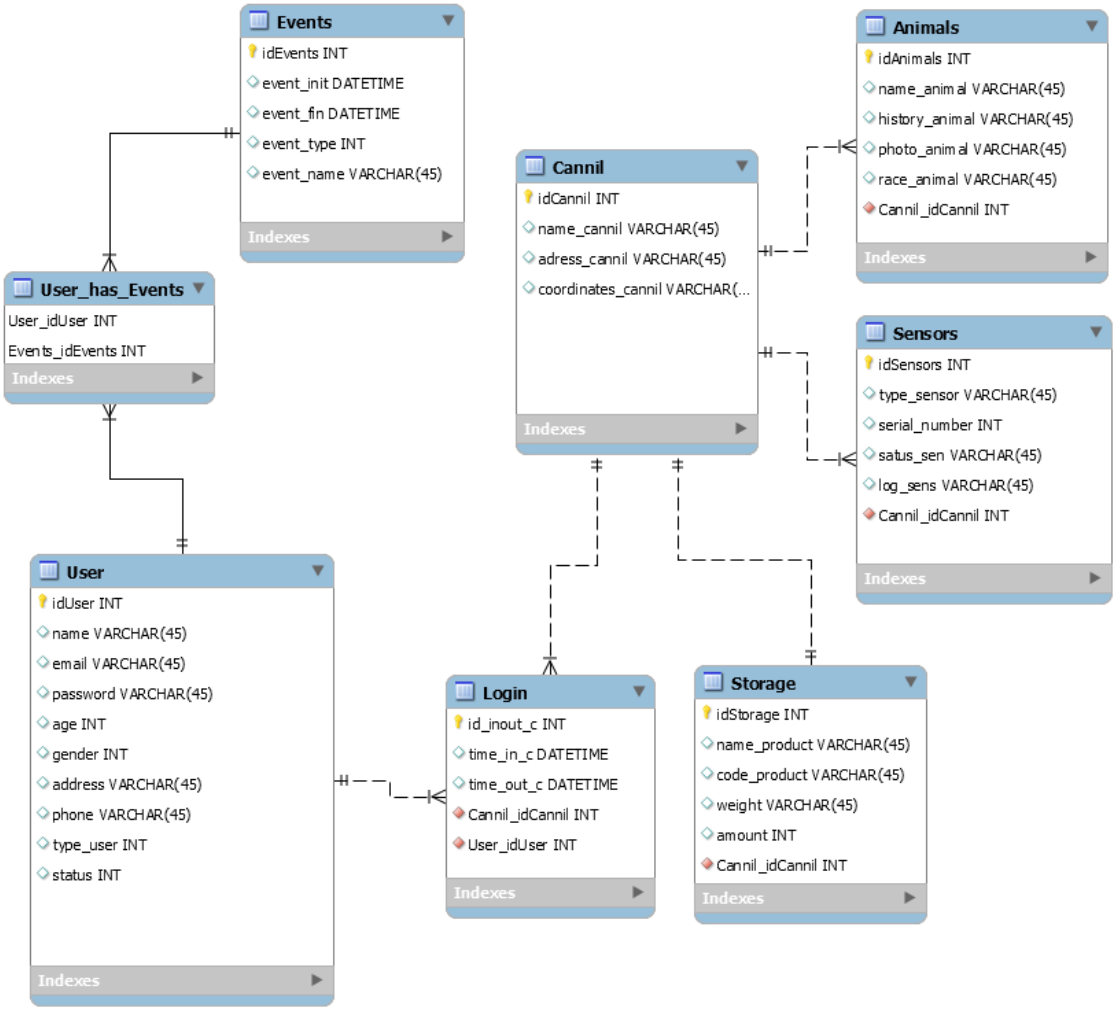
O utlizador pode ver as informações dos animais do canil em que está. Na classe **"Animals"** são registados os nomes dos animais, o histórico, foto e raça. Estas informações estão implicitas nos atributos: **name_animal**, **history_animal**, **photo_animal** e **race_animal**.

O utilizador também pode verificar o estado dos sensores em tempo real. A classe **"Sensors"** é a responsável por isso, sendo que ela regista o tipo de sensor, o seu serial number, o seu estado e os logs. Estas informações estão implicitas nos atributos: **type_sensor**, **serial_number**, **satus_sensor** e **log_sensor**.

Por último, o utilizador também pode verificar o stock presente no armazém e fazer a sua atualização. A classe **"Storage"** regista o nome dos produtos, o seu código, peso e quantidade. Estas informações estão implicitas nos atributos: **name_product**, **code_product**, **weight** e **amount**.



6 Conversão do Modelo de entidades para o Modelo Relacional



7 Criação de Tabelas:

De acordo com o modelo relacional é gerado o seguinte código SQL para criação das tabelas. Este processo é automatizado através do software Mysql Workbench. Nestas tabelas são utilizadas chaves primárias e chaves estrangeiras de modo a podermos aceder às várias tabelas conforme o especificado no modelo relacional.

```
1  -- MySQL Script generated by MySQL Workbench
2  -- Mon Mar 22 19:21:55 2021
3  -- Model: New Model    Version: 1.0
4  -- MySQL Workbench Forward Engineering
5
6  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
7  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
       FOREIGN_KEY_CHECKS=0;
8  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,
       STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,
       ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
9
10 -- -----------------------------------------------------
11 -- Schema logipeet
12 -- -----------------------------------------------------
13
14 -- -----------------------------------------------------
15 -- Schema logipeet
16 -- -----------------------------------------------------
17 CREATE SCHEMA IF NOT EXISTS `logipeet` DEFAULT CHARACTER SET
       utf8 ;
18 USE `logipeet` ;
19
20 -- -----------------------------------------------------
21 -- Table `logipeet`.`User`
22 -- -----------------------------------------------------
23 CREATE TABLE IF NOT EXISTS `logipeet`.`User` (
24   `idUser` INT NULL AUTO_INCREMENT,
25   `name` VARCHAR(45) NULL,
26   `email` VARCHAR(45) NULL,
27   `password` VARCHAR(45) NULL,
28   `age` INT NULL,
```

```
  `gender` INT NULL,
  `address` VARCHAR(45) NULL,
  `phone` VARCHAR(45) NULL,
  `type_user` INT NULL,
  `status` INT NULL,
  PRIMARY KEY (`idUser`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `logipeet`.`Events`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `logipeet`.`Events` (
  `idEvents` INT NOT NULL,
  `event_init` DATETIME NULL,
  `event_fin` DATETIME NULL,
  `event_type` INT NULL,
  `event_name` VARCHAR(45) NULL,
  PRIMARY KEY (`idEvents`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `logipeet`.`Cannil`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `logipeet`.`Cannil` (
  `idCannil` INT NOT NULL AUTO_INCREMENT,
  `name_cannil` VARCHAR(45) NULL,
  `adress_cannil` VARCHAR(45) NULL,
  `coordinates_cannil` VARCHAR(45) NULL,
  PRIMARY KEY (`idCannil`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `logipeet`.`Storage`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `logipeet`.`Storage` (
  `idStorage` INT NOT NULL AUTO_INCREMENT,
  `name_product` VARCHAR(45) NULL,
  `code_product` VARCHAR(45) NULL,
  `weight` VARCHAR(45) NULL,
  `amount` INT NULL,
  `Cannil_idCannil` INT NOT NULL,
  PRIMARY KEY (`idStorage`),
  INDEX `fk_Storage_Cannil1_idx` (`Cannil_idCannil` ASC)
   VISIBLE,
  CONSTRAINT `fk_Storage_Cannil1`
    FOREIGN KEY (`Cannil_idCannil`)
```

```
    REFERENCES `logipeet`.`Cannil` (`idCannil`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `logipeet`.`Login`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `logipeet`.`Login` (
  `id_inout_c` INT NOT NULL AUTO_INCREMENT,
  `time_in_c` DATETIME NULL,
  `time_out_c` DATETIME NULL,
  `Cannil_idCannil` INT NOT NULL,
  `User_idUser` INT NOT NULL,
  PRIMARY KEY (`id_inout_c`),
  INDEX `fk_Login_Cannil1_idx` (`Cannil_idCannil` ASC)
   VISIBLE,
  INDEX `fk_Login_User1_idx` (`User_idUser` ASC) VISIBLE,
  CONSTRAINT `fk_Login_Cannil1`
    FOREIGN KEY (`Cannil_idCannil`)
    REFERENCES `logipeet`.`Cannil` (`idCannil`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Login_User1`
    FOREIGN KEY (`User_idUser`)
    REFERENCES `logipeet`.`User` (`idUser`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `logipeet`.`Animals`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `logipeet`.`Animals` (
  `idAnimals` INT NOT NULL AUTO_INCREMENT,
  `name_animal` VARCHAR(45) NULL,
  `history_animal` VARCHAR(45) NULL,
  `photo_animal` VARCHAR(45) NULL,
  `race_animal` VARCHAR(45) NULL,
  `Cannil_idCannil` INT NOT NULL,
  PRIMARY KEY (`idAnimals`),
  INDEX `fk_Animals_Cannil1_idx` (`Cannil_idCannil` ASC)
   VISIBLE,
  CONSTRAINT `fk_Animals_Cannil1`
    FOREIGN KEY (`Cannil_idCannil`)
    REFERENCES `logipeet`.`Cannil` (`idCannil`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `logipeet`.`Sensors`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `logipeet`.`Sensors` (
  `idSensors` INT NOT NULL AUTO_INCREMENT,
  `type_sensor` VARCHAR(45) NULL,
  `serial_number` INT NULL,
  `satus_sen` VARCHAR(45) NULL,
  `log_sens` VARCHAR(45) NULL,
  `Cannil_idCannil` INT NOT NULL,
  PRIMARY KEY (`idSensors`),
  INDEX `fk_Sensors_Cannil1_idx` (`Cannil_idCannil` ASC)
   VISIBLE,
  CONSTRAINT `fk_Sensors_Cannil1`
    FOREIGN KEY (`Cannil_idCannil`)
    REFERENCES `logipeet`.`Cannil` (`idCannil`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `logipeet`.`User_has_Events`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `logipeet`.`User_has_Events` (
  `User_idUser` INT NOT NULL,
  `Events_idEvents` INT NOT NULL,
  PRIMARY KEY (`User_idUser`, `Events_idEvents`),
  INDEX `fk_User_has_Events_Events1_idx` (`Events_idEvents`
   ASC) VISIBLE,
  INDEX `fk_User_has_Events_User1_idx` (`User_idUser` ASC)
   VISIBLE,
  CONSTRAINT `fk_User_has_Events_User1`
    FOREIGN KEY (`User_idUser`)
    REFERENCES `logipeet`.`User` (`idUser`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_User_has_Events_Events1`
    FOREIGN KEY (`Events_idEvents`)
    REFERENCES `logipeet`.`Events` (`idEvents`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


```

```
170 SET SQL_MODE=@OLD_SQL_MODE;
171 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
172 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```