MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECT

la disciplina
Baze de Date

Rentbike

Alexandra-Ioana Varga, 30223
An academic : 2020-2021

Cuprins

- Elemente de utilizare/instalare (diferențiat pe tipuri de actori);
- Elemente de securizare a aplicației;

5. Concluzii, limitări și dezvoltări ulterioare

6. Bibliografie

1. Introducere

❖ **Introducere, argumente, scop și obiective specifice**

- În ziua de astăzi, din ce în ce mai multe firme doresc să se afirme în mediul online, acestea oferind o metodă de extindere a razei de clienți, dar și o cale mai rapidă de a realiza comunicarea dintre client și firmă.

- Astfel, proiectul realizat transpune în mediul online aplicația unei firme ce se ocupă cu închirierea bicicletelor. Acesta oferă utilizatorului posibilitatea realizării unei închirieri mai ușoare, scutindu-l de drumul până la locația firmei și de timpul petrecut la coadă. Totodată prin utilizarea unei baze de date ce stochează informațiile despre clienți și rezervări, procesul se ușurează și pentru angajații firmei, o mare parte din stocare realizându-se automat.

- Utilizând aplicația, clientul are bonus posibilitatea de a vizualiza bicicletele în cadrul ofertelor disponibile.

2. Analiza cerințelor utilizatorilor (Specificațiile de proiect)

❖ **Ipoteze specifice domeniului ales pentru proiect (cerințe, constrângeri)**

- Mediul online dispune de o mulțime de site-uri cu diferite scopuri, fiecare remarcându-se într-un mod unic. Scopul principal al unui astfel de mijloc de comunicare este de a realiza transmiterea informațiilor cât mai ușor către utilizator. Metodele sunt multiple iar parametrii ce trebuie luați în calcul sunt de asemenea numeroși. Unul dintre cei mai importanți este design-ul. Acesta trebuie să fie atractiv, dar să nu distragă clientul de la a-și realiza scopul, să fie armonios dar nu prea banal.

- Culorile și formele trebuie alese cu grijă pentru a nu deranja vizual, iar așezarea lor trebuie să fie strategică pentru a crea o ambianța plăcută. Un design bine realizat va face navigarea în cadrul site-ului mult mai ușoară.

- Un al parametru ce trebuie luat în calcul este prezentarea informației. Pentru ca utilizatorul să primească informațiile într-un mod eficient, aceastea trebuie să fie dispuse structurat, pe categorii.

❖ **Organizare structurată(tabelar) a cerințelor utilizator;**

Baza de date conține informațiile organizate în tabele, astfel avem tabele ce stochează informațiile despre :

- biciclete(model, tip, producător etc.);
- clienți(nume, prenume, username etc.);
- rezervări(username, bicicletă, dată etc.);
- locații(nume etc.);
- oferte(bicicletă, valabilitate etc,);

❖ **Determinarea și caracterizarea de profiluri de utilizatori (admin, user , diverși alți "actori")**

Aplicația permite 3 tipuri de utilizatori, fiecare având anumite opțiuni disponibile, în funcție de nivelul de acces permis:

- Vizitator - utilizatorul nelogat;
- Client - utilizator logat;
- Administrator - utilizator logat cu un tip diferit de acces;

În calitate de vizitator, un utilizator are permisiunea:

- să își creeze un cont, introducând informațiile necesare;

- să se logheze, folosind username-ul și parola alese la crearea contului, ajungând astfel în rolul de client;
- să modifice datele contului sau parola;

În calitate de client, un utilizator are permisiunea:

- să analizeze ofertele disponibile ale bicicletelor;
- să creeze o rezervare a unei biciclete, alegând bicicleta, locul de unde dorește să o închirieze și data zilei respective;
- să verifice disponibilitatea de rezervare a unei biciclete, alegând bicicleta, locul de unde dorește să o închirieze și data zilei respective;
- să anuleze o rezervare deja făcută a unei biciclete;
- să se delogheze de pe aplicație, revenind la nivelul de vizitator;

În calitate de administrator, un utilizator are permisiunea:

- să adauge un nou tip de biciclete la baza de date;
- să vizualizeze istoricul rezervărilor;
- să vizualizeze clienții bazei de date a magazinului;
- să vizualizeze rapoarte despre închirierile bicicletelor;
- să se delogheze de pe aplicație, revenind la nivelul de vizitator;

Aceste funcționalități sunt implementate la nivelul aplicației pentru o gestionare mai ușoară a bicicletelor, însă administratorul are de fapt posibilitatea de a șterge, updata sau vedea orice informație din baza de date.

3. Modelul de date și descrierea acestuia

- **Entități și atributele lor (descriere detaliată – implementarea fizică)**
  Baza de date conține 11 tabele, fiecare cu atributele specifice.

- **Tabela client**

În acest tabel sunt salvate datele clienților care și-au creat cont pe aplicație. Câmpurile sunt: CNP(PK), Nume, Prenume, Telefon, Email, IDdate_logare(FK), IDadresa(FK), Cont_Bancar(FK).

- **Tabela adresa**

Acest tabel conține adresele clienților dar și a magazinelor de unde se pot prelua bicicletele închiriate. Câmpurile sunt: idAdresa (PK), Oraș, Număr, Stradă, Cod_postal.

- **Tabela date_logare**

În acest tabel au fost introduse date de logare pentru aplicație, ale clienților. Acesta conține: idDate_logare(PK), Username, Parola.

- **Tabela Cont**

Acest tabel conține modul de plată al clienților. Câmpurile sunt: Cont(PK), Data_expirare, Cod_siguranta, tip.

- **Tabela bicicleta**

Memorează datele despre bicicletele care pot fi închiriate. Acesta conține: idBicicleta(PK), Producător , Model, IDcaracteristici(FK), TARIF.

- **Tabela caracteristici**

În acest tabel sunt memorate caracteristicile fiecărei biciclete. Câmpurile sunt: idCaracteristici(PK), Culoare, Nr_viteze, Tip_frana, Diametru_roata, IDtip(FK).

- **Tabela tip**

Memorează tipul de bicicletă (mountain bike, city bike, children bike, road bike).

- **Tabela rent**

În acest tabel sunt salvate toate închirierile efectuate de client. Acesta conține: idRent(PK), IDbicicleta(FK), CNP_client(FK), IDperioada(FK), IDlocatie(FK), Data_rezervarii, Pret_total.

- **Tabela perioada**

Acest tabel conține date referitoare la data când sunt efectuate închirierile. Câmpurile sunt: idPerioada(PK), Dată, Ora_inceput, Ora_sfarsit.

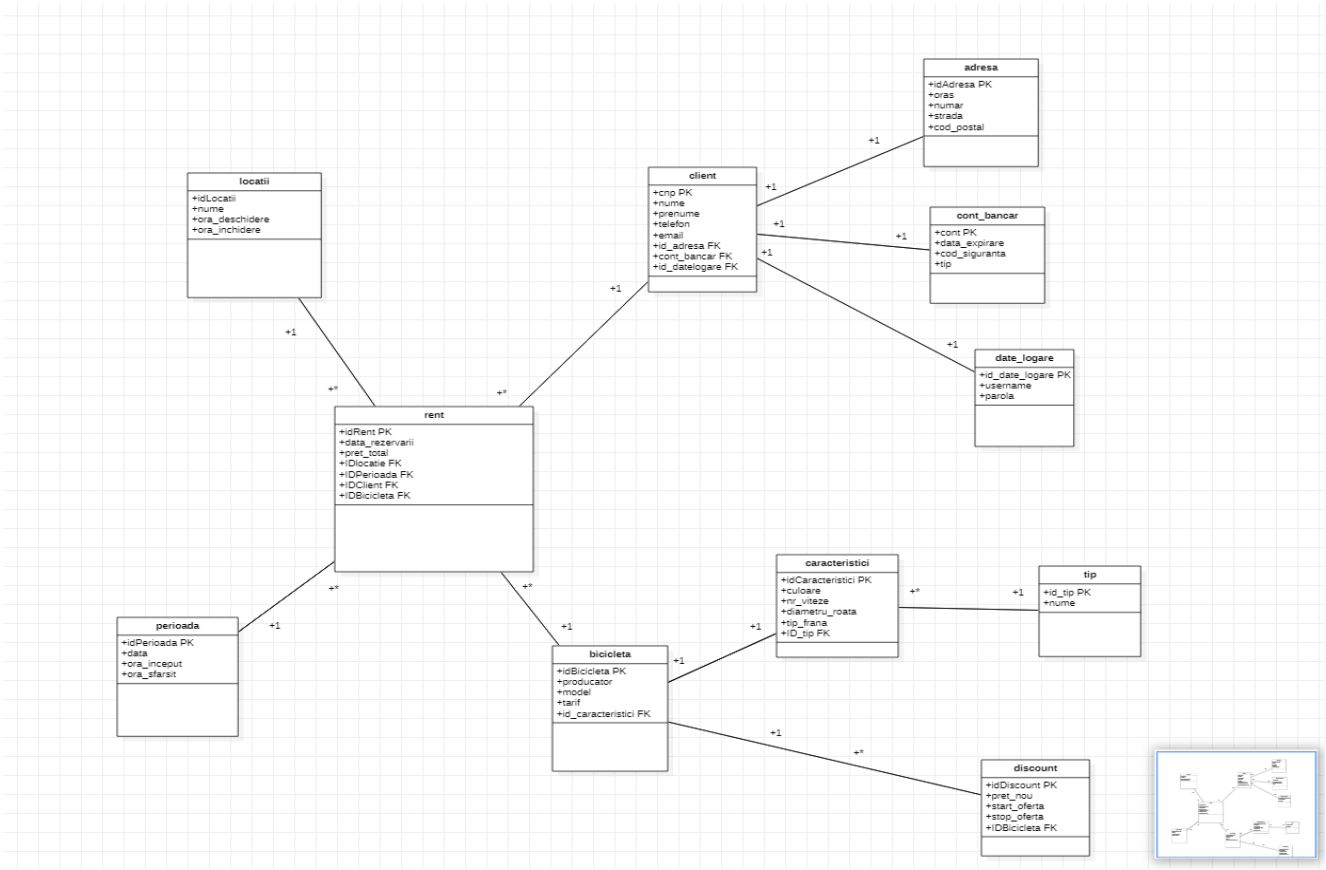
- **Tabela locatii**

În acest tabel sunt introduse datele despre magazinele de unde se pot închiria bicicletele. Acesta conține: idLocatii(PK), Nume, Adresa_ID(FK), Ora_deschidere, Ora_inchidere.

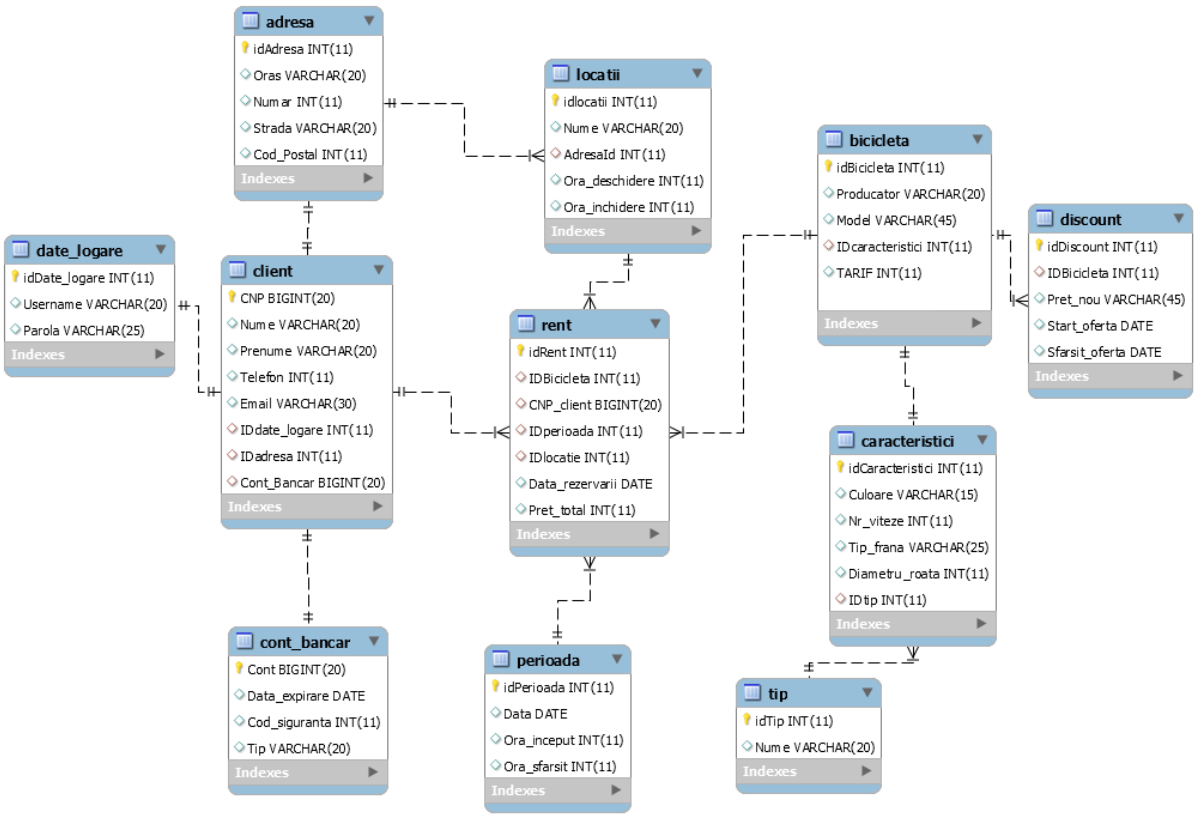
- **Tabela discount**

Acest tabel conține informații despre bicicletele care sunt la oferta într-o anumită perioada. Câmpurile sunt: IdDiscount(PK), IDbicicleta(FK), Pret_nou, Start_oferta, Sfarsit_oferta.

❖ **Diagrama UML pentru modelul de date complet**



❖ **Diagrama EER pentru modelul de date complet**



❖ **Normalizarea datelor**

Când proiectăm o bază de date, dorința noastră este de a reprezenta cât mai corect informațiile și să dimininuăm cât mai mult posibilitatea de a ajunge la informații eronate. Pentru a ajunge la această perforamanță, trebuie să folosim normalizarea. Procesul de normalizare este o metodă formală de identificare a relaţiilor bazate pe chei primare (sau chei candidat în cazul BCNF) şi dependenţele funcţionale cu atributele asociate.

- **Prima formă de normalizare (1NF)**

Schema se află în prima formă normală dacă și numai dacă toate atributele sale iau valori atomice.

De exemplu:

Idbicicleta	**Nume**
1	Giant speedx
2	Pegas beary

Numele conține producătorul bicicletei și modelul deci pentru a fi în 1NF am despărțit în două tabele:

IDbicicleta	**Producător**	**idModel**
1	Giant	1
2	Pega	2

idModel	**Model**
1	Speedx
2	beary

- **A două formă normală (2NF)**

O relație este în a doua formă normală dacă și numai dacă se găsește în prima formă normală și în plus orice atribut ce nu este cheie primară este dependent total de cheia primară. Pentru a asigura dependența față de întreaga cheie primară, este necesar ca în tabelele care au cheie compusă, fiecare atribut să depindă de toate coloanele care compun cheia primară. Dacă un tabel are cheie unică el intră automat în 2NF.

De exemplu: în cazul tabelei ‘Perioada’, cheia primară care ne asigură unicitatea este reprezentată de câmpul ‘idPerioada’, restul câmpurilor (‘Data’, ‘Data_inceput’, ‘Data_final’) depind total de aceasta.

- **A treia formă normală (3NF)**

O relație este în a treia formă normală dacă este în FN2 și nici un atribut care nu este cheie primară nu este dependent funcțional de un alt atribut care nu este cheie primară.

De exemplu: În tabela 'rent ' nu s-a introdus și numele clientului care face rezervarea deoarece acesta poate fi dedus din câmpul 'CNP_client'.

- **Forma normală Boyce-Codd**

O relație este în forma normală BCNF dacă este în FN3 și orice atribut este relație / cheie candidat. Singurele dependențe funcționale permise sunt cele în care o cheie determină orice alt atribut. Din păcate, această baza de date nu a putut fi adusă în această formă normală. [5]

❖ **Relații**

✓ ***One-to-one* (1-1)**

Unei înregistrări din tabela părinte îi corespunde o înregistrare în tabela copil (de exemplu, unui client îi corespunde o singură adresă de domiciliu).

În această baza de date se pot regăsi mai multe de relații de unu la unu.

De exemplu: între tabela "client" și "date_logare" există o relație de unu la unu, fiecărui client îi corespunde un singur username și o singură parolă, acestea fiind necesare pentru a se conecta în aplicație și a putea fi identificat unic.

✓ ***One-to-many* (1-N)**

Unei înregistrări din tabela părinte îi corespund mai multe înregistrări în tabela copil (de exemplu, un department poate avea mai multe proiecte).

În cadrul acestei baze de date există câteva relații 1-N. De exemplu, între tabela "client" și "rent" există o astfel de relație în sensul că orice client dacă dorește poate să facă mai

multe rezervări pentru aceeași bicicletă sau pentru biciclete diferite. De asemenea, între tabela "bicicleta" și "discount" există o relație 1-N, fiecare bicicletă poate avea unul sau mai multe discount-uri pentru perioade de timp diferite.

- ✓ ***Many- to-many (N-N)*** : Acest tip de relații sunt modelate în baza de date prin crearea unei tabele de legătură.

De exemplu: între tabelele "client" și "bicicleta" există o relație N-N deoarece un client poate închiria mai multe biciclete și o bicicletă poate fi închiriată de mai mulți clienți. Această relație a fost împărțită în două relații de 1-N prin introducerea tabelei 'rent'.[6]

4. Detalii de implementare

- ❖ **Descrierea funcțională a modulelor (organizarea logică a acestora)**
- ❖ **Tool-uri**

MySQL WorkBench

- MySQL WorkBench este un instrument grafic pentru a lucra cu serverele și bazele de date MySQL. Acesta este o aplicație ce acoperă cele mai importante activități de gestionare a bazelor de date:
- SQL Development: permite să gestioneze conexiunile la serverele MySQL, de asemenea oferă posibilitatea de a executa interogări SQL;
- Data Modeling: permite să creezi modele de baze de date în mod grafic, cât și editarea de date deja existente;
- Server Administration: permite să creezi și să administrezi serverul MySQL;
- Data Migration: permite să tranferi în MySQL date din Microsoft SQL Server, Sybase ASE, SQLite, SQL Anywhere, PostgreSQl; [3]

IDEA IntelliJ

- IntelliJ IDEA este un mediu de dezvoltare integrat (IDE) scris în Java pentru dezvoltarea de software. Este dezvoltat de JetBrains (cunoscut anterior ca IntelliJ) și

este disponibil ca ediție comunitară licențiată Apache 2 , și într-o ediție comercială proprietară.

- IDE oferă anumite caracteristici cum ar fi completarea codului prin analizarea contextului, navigarea prin cod care permite săriți direct la o clasă sau o declarație în cod, refactoring-ul codului , depanarea codului și opțiuni pentru a remedia neconcordanțele prin sugestii.
- IDE oferă integrarea cu instrumente de construcție/ ambalare, cum ar fi grunt , bower, gradle și SBT . Suportă sisteme de control al versiunilor precum Git , Mercurial , Perforce și SVN . Bazele de date precum Microsoft SQL Server , Oracle , PostgreSQL , SQLite și MySQL pot fi accesate direct din IDE în ediția Ultimate, printr-o versiune încorporată a DataGrip.[2]

❖ Limbaje de programare

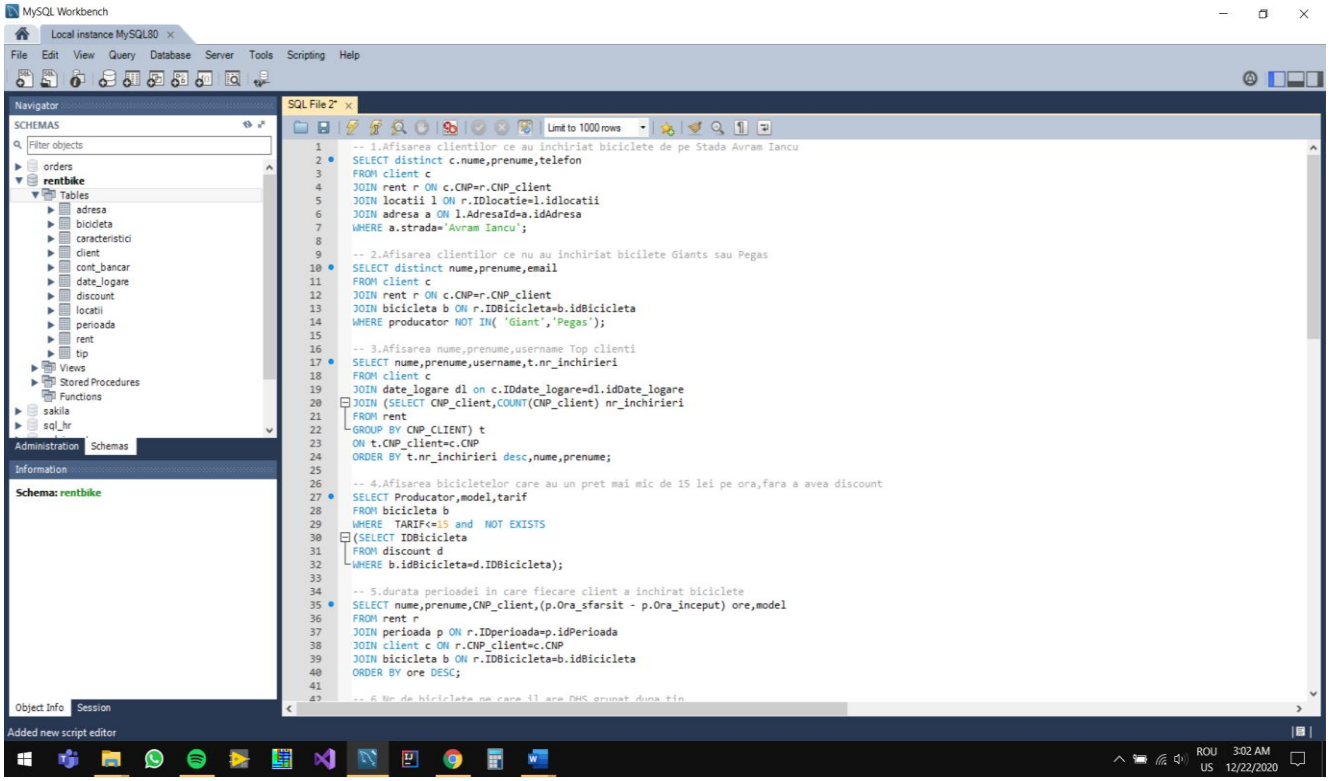
➢ SQL

SQL (Structured Query Language) este un limbaj de programare specific pentru manipularea datelor în sistemele bazelor de date relaționale (RDBMS), iar la origine este un limbaj bazat pe algebra relațională. Acesta are scop inserarea datelor, interogări, actualizări, ștergere, modificare și crearea schemelor. A devenit un standard în domeniu, fiind cel mai popular limbaj utilizat pentru manipularea datelor pentru SGBD-urile relaționale.[4]

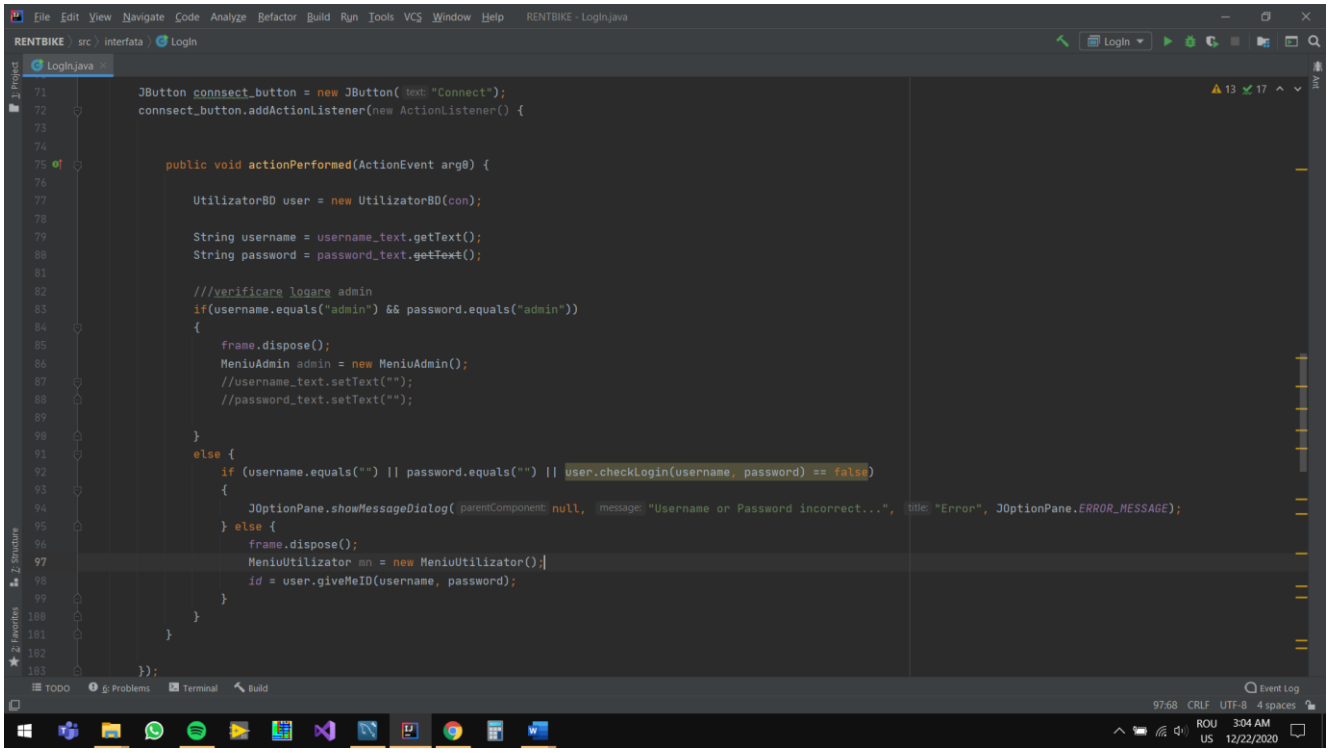
➢ Java

Java este un limbaj de programare orientat-obiect, puternic tipizat, iar noile evoluții tehnologice permit utilizarea sa și pe dispozitive mobile, spre exemplu telefon, agenda electronică etc. Acesta este utilizat în prezent cu succes și pentru programarea aplicațiilor destinate intranet-urilor. Mașina virtuală Java este mediul în care se execută programele Java. În prezent, există mai mulți furnizori de JVM, printre care Oracle, IBM, Bea, FSF.[1]

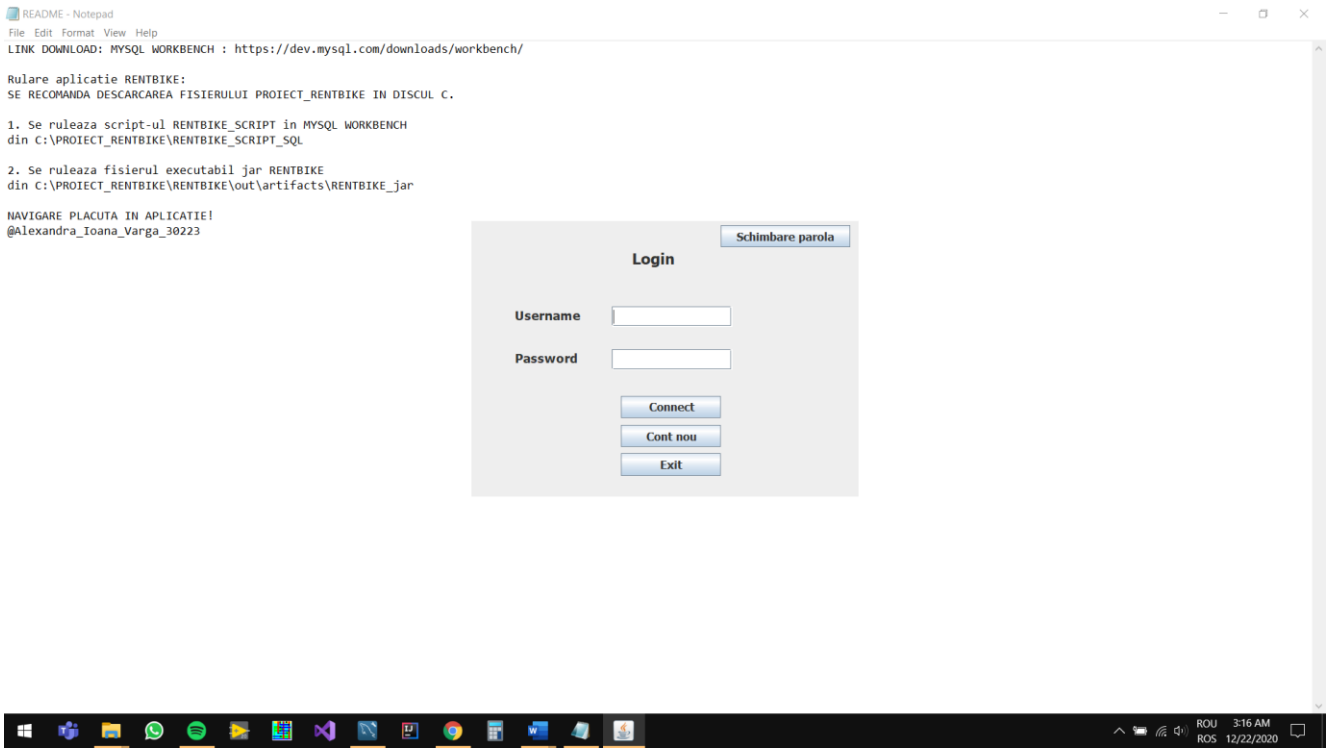
Exemplu de interogări ale bazei de date scrise în SQL:



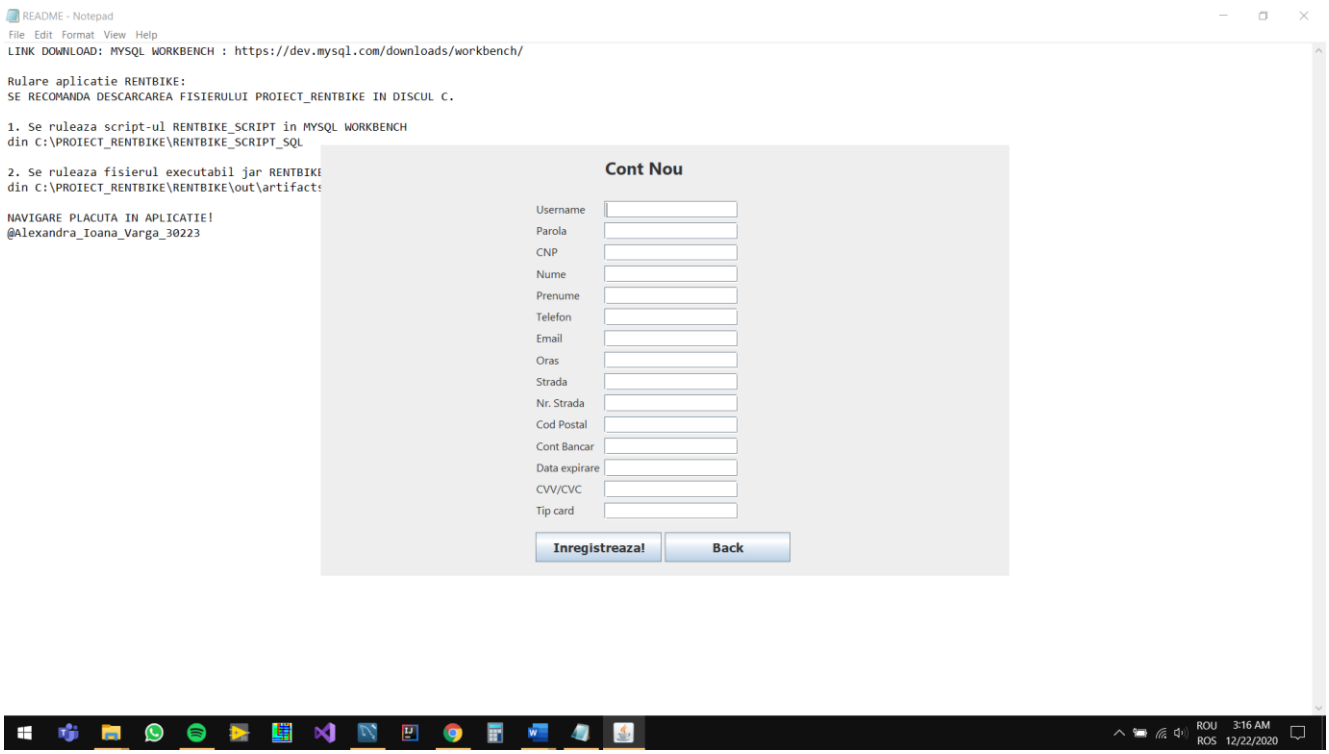
Exemplu clasă a interfeței aplicației bazei de date scrise în Java:

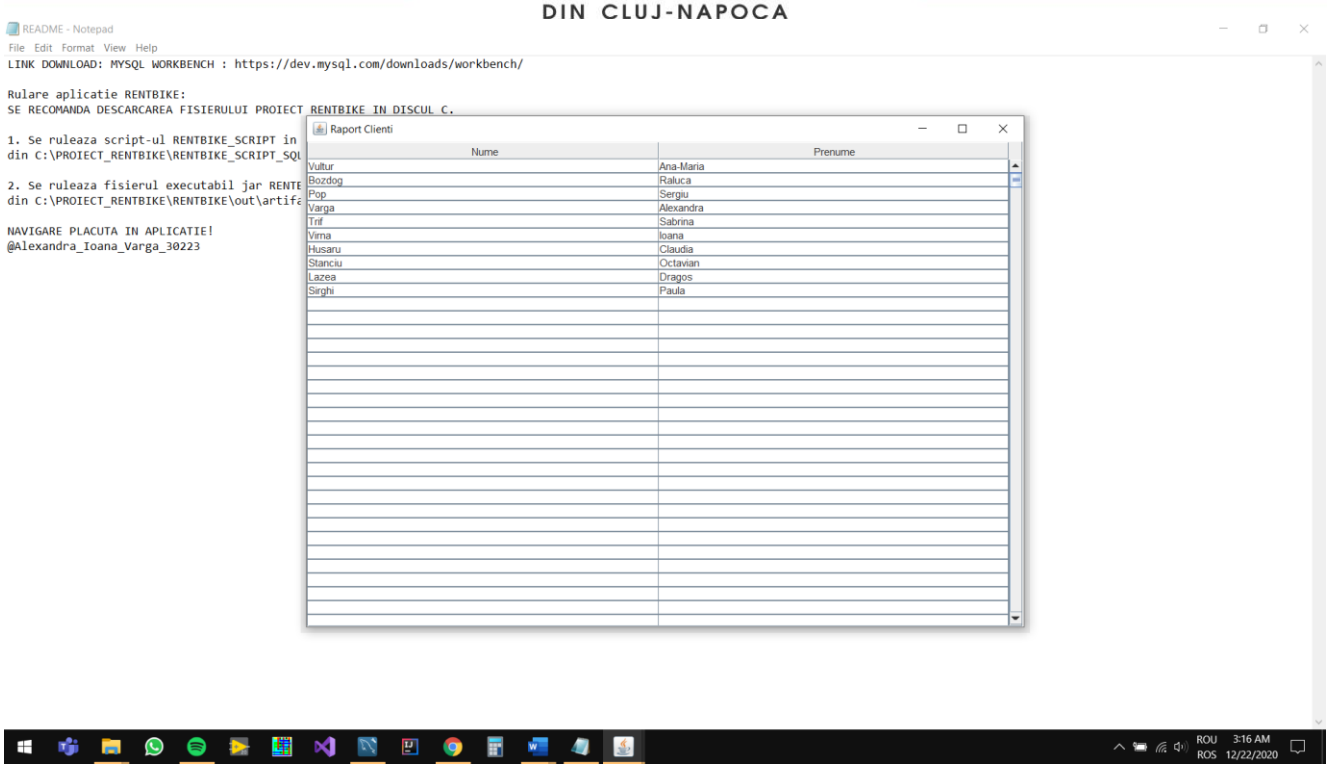


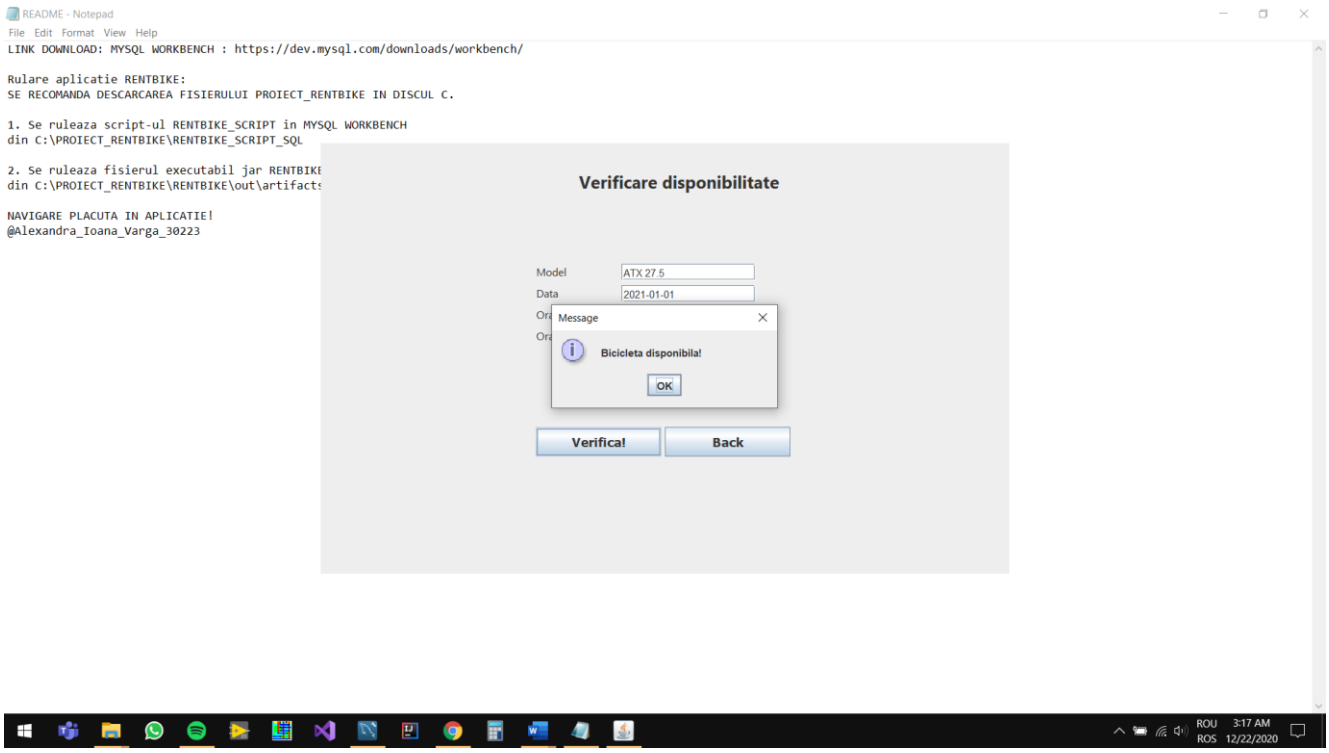
❖ **Elemente de utilizare/instalare (diferențiat pe tipuri de actori)**



Exemple navigare aplicație:







REZUMAT NAVIGARE:

1. **SE EXECUTĂ PAȘII DE ACCESARE A APLICAȚIEI CONFORM FIȘIERULUI README.**

2. **DUPĂ LANSAREA FERESTREI DE LOGIN POATE FI TESTATĂ APLICAȚIA DE CĂTRE UTILIZATOR.**

❖ **Elemente de securizare a aplicației**

Securitatea aplicației este realizată prin 2 moduri: folosirea sesiunilor și conectarea la baza de date în funcție de utilizator. Variabilele de sesiune sunt utilizare pentru menținerea datelor unui utilizator atunci când este logat, ele fiind distruse atunci când el se delogheaza. Conectarea la baza de date se face diversificat pentru fiecare tip de utilizator, astfel vizitatorii au accesul cel mai redus. Clientul are un nivel de acces mai ridicat, el putând să își vadă informațiile despre cont și să le modifice. Administratorul are accesul limitat doar de performanțele bazei de date.

5. Concluzii, limitări și dezvoltări ulterioare

Integrând aplicația în mediu online (mobil), oferim oportunitatea clienților de a-și ușura căutarea unei biciclete de închiriat, iar angajaților firmei le oferim o metodă structurată și ușoară de gestionare a rezervărilor.

Aplicația fiind încă la început, aceasta permite o gamă largă de dezvoltări ulterioare ce ar putea îmbunătăți calitatea timpului petrecut în aplicație al utilizatorilor.

6. Bibliografie

- Anexă cod SQL – RENTBIKE_SCRIPT
- https://ro.wikipedia.org/wiki/Java_(limbaj_de_programare) [1]
- https://en.wikipedia.org/wiki/IntelliJ_IDEA [2]
- https://www.mysql.com/products/workbench/ [3]
- https://ro.wikipedia.org/wiki/SQL [4]
- https://ftp.utcluj.ro/pub/users/civan/IBD/2_LABORATOR/08_Normalizare/ [5]
- https://ftp.utcluj.ro/pub/users/civan/IBD/2_LABORATOR/02_SQL_ModelulRelational/ [6]

ANEXA COD SQL

#CREARE TABELE

-- hack pentru

-- ERROR 1452: Cannot add or update a child row: a foreign key constraint fails

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=1;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=1;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- drop schema rentbike;

CREATE SCHEMA IF NOT EXISTS `rentbike`;

USE `rentbike` ;

-- tabela date logare

-- clientul va trebui sa creeze un cont in cadrul magazinului de inchirieri biciclete

```
CREATE TABLE IF NOT EXISTS `rentbike`.`Date_logare` (
  `idDate_logare` INT NOT NULL AUTO_INCREMENT,
  `Username` VARCHAR(20) NULL,
  `Parola` VARCHAR(25) NULL,
  PRIMARY KEY (`idDate_logare`))
ENGINE = InnoDB;


-- tabela adresa
-- contine informatii despre adresa clientului
CREATE TABLE IF NOT EXISTS `rentbike`.`Adresa` (
  `idAdresa` INT NOT NULL AUTO_INCREMENT,
  `Oras` VARCHAR(20) NULL,
  `Numar` INT NULL,
  `Strada` VARCHAR(20) NULL,
  `Cod_Postal` INT NULL,
  PRIMARY KEY (`idAdresa`))
ENGINE = InnoDB;


-- tabela cont bancar
-- contine informatii despre cardul clientului
```

```
CREATE TABLE IF NOT EXISTS `rentbike`.`Cont_Bancar` (
  `Cont` BIGINT NOT NULL,
  `Data_expirare` DATE NULL,
  `Cod_siguranta` INT NULL,
  `Tip` VARCHAR(20) NULL,
  PRIMARY KEY (`Cont`))
ENGINE = InnoDB;


-- tabela client
-- contine datele despre clientul inrolat in baza de date
-- a magazinului de biciclete


CREATE TABLE IF NOT EXISTS `rentbike`.`Client` (
  `CNP` BIGINT NOT NULL,
  `Nume` VARCHAR(20) NULL,
  `Prenume` VARCHAR(20) NULL,
  `Telefon` INT NULL,
  `Email` VARCHAR(30) NULL,
  `IDdate_logare` INT NULL,
```

```
`IDadresa` INT NULL,
`Cont_Bancar` BIGINT NULL,
PRIMARY KEY (`CNP`),
INDEX `logara_idx` (`IDdate_logare` ASC),
INDEX `adresa_idx` (`IDadresa` ASC),
INDEX `cont_idx` (`Cont_Bancar` ASC),
CONSTRAINT `logare`
  FOREIGN KEY (`IDdate_logare`)
  REFERENCES `rentbike`.`Date_logare` (`idDate_logare`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `adresa`
  FOREIGN KEY (`IDadresa`)
  REFERENCES `rentbike`.`Adresa` (`idAdresa`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `cont`
  FOREIGN KEY (`Cont_Bancar`)
  REFERENCES `rentbike`.`Cont_Bancar` (`Cont`)
  ON DELETE NO ACTION
```

```
  ON UPDATE NO ACTION)

ENGINE = InnoDB;



-- tabela tip

-- contine date despre tipul de biciclete existente la magazin


CREATE TABLE IF NOT EXISTS `rentbike`.`Tip` (

  `idTip` INT NOT NULL AUTO_INCREMENT,

  `Nume` VARCHAR(20) NULL,

  PRIMARY KEY (`idTip`))

ENGINE = InnoDB;



-- -----------------------------------------------------

-- Table `rentbike`.`Caracteristici`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `rentbike`.`Caracteristici` (

  `idCaracteristici` INT NOT NULL AUTO_INCREMENT,

  `Culoare` VARCHAR(15) NULL,
```

```
  `Nr_viteze` INT NULL,
  `Tip_frana` VARCHAR(25) NULL,
  `Diametru_roata` INT NULL,
  `IDtip` INT NULL,

  PRIMARY KEY (`idCaracteristici`),
  INDEX `tip_idx` (`IDtip` ASC),
  CONSTRAINT `tip`
    FOREIGN KEY (`IDtip`)
    REFERENCES `rentbike`.`Tip` (`idTip`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- tabela biciclete contine date despre
-- indentificarea bicicletei producator, model, tarif

CREATE TABLE IF NOT EXISTS `rentbike`.`Bicicleta` (
  `idBicicleta` INT NOT NULL AUTO_INCREMENT,
  `Producator` VARCHAR(20) NULL,
```

```
  `Model` VARCHAR(45) NULL,
  `IDcaracteristici` INT NULL,
  `TARIF` INT NULL,
  PRIMARY KEY (`idBicicleta`),
  INDEX `caracteristici_idx` (`IDcaracteristici` ASC),
  CONSTRAINT `caracteristici`
    FOREIGN KEY (`IDcaracteristici`)
    REFERENCES `rentbike`.`Caracteristici` (`idCaracteristici`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- tabela perioada contine date despre
-- perioada in care s-au efectuat inchirieri ale bicicletelor
-- trebuie sa avem grija ca inchirierile sa nu se suprapuna


CREATE TABLE IF NOT EXISTS `rentbike`.`Perioada` (
  `idPerioada` INT NOT NULL AUTO_INCREMENT,
  `Data` DATE NULL,
  `Ora_inceput` INT NULL,
```

```
  `Ora_sfarsit` INT NULL,
  PRIMARY KEY (`idPerioada`))
ENGINE = InnoDB;



-- tabela discount
-- contine diverse oferte pentru anumite bicicicletе

CREATE TABLE IF NOT EXISTS `rentbike`.`Discount` (
  `idDiscount` INT NOT NULL AUTO_INCREMENT,
  `IDBicicleta` INT NULL,
  `Pret_nou` VARCHAR(45) NULL,
  `Start_oferta` DATE NULL,
  `Sfarsit_oferta` DATE NULL,
  PRIMARY KEY (`idDiscount`),
  INDEX `bic_idx` (`IDBicicleta` ASC),
  CONSTRAINT `bic`
    FOREIGN KEY (`IDBicicleta`)
    REFERENCES `rentbike`.`Bicicleta` (`idBicicleta`)
    ON DELETE NO ACTION
```

```
  ON UPDATE NO ACTION)

ENGINE = InnoDB;


-- tabela locatii

-- contine informatii despre locatiile din care

-- se pot inchiria biciclete


CREATE TABLE IF NOT EXISTS `rentbike`.`Locatii` (

  `idlocatii` INT NOT NULL AUTO_INCREMENT,

  `Nume` VARCHAR(25) NULL,

  `AdresaId` INT NULL,

  `Ora_deschidere` INT NULL,

  `Ora_inchidere` INT NULL,

  PRIMARY KEY (`idlocatii`),

  INDEX `asd_idx` (`AdresaId` ASC),

  CONSTRAINT `asd`

    FOREIGN KEY (`AdresaId`)

    REFERENCES `rentbike`.`Adresa` (`idAdresa`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;



-- tabela rent

-- este tabela de rezevari in care se retin datele clientilor

-- care au rezervat biciclete


CREATE TABLE IF NOT EXISTS `rentbike`.`Rent` (

  `idRent` INT NOT NULL AUTO_INCREMENT,

  `IDBicicleta` INT NULL,

  `CNP_client` BIGINT NULL,

  `IDperioada` INT NULL,

  `IDlocatie` INT NULL,

  `Data_rezervarii` DATE NULL,

  `Pret_total` INT NULL,

  PRIMARY KEY (`idRent`),

  INDEX `bicicleta_idx` (`IDBicicleta` ASC),

  INDEX `client_idx` (`CNP_client` ASC),

  INDEX `perioada_idx` (`IDperioada` ASC),

  CONSTRAINT `bicicleta1`
```

```
  FOREIGN KEY (`IDBicicleta`)
  REFERENCES `rentbike`.`Bicicleta` (`idBicicleta`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `client1`
  FOREIGN KEY (`CNP_client`)
  REFERENCES `rentbike`.`Client` (`CNP`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `locatie1`
  FOREIGN KEY (`IDlocatie`)
  REFERENCES `rentbike`.`locatii` (`IDlocatii`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `perioada1`
  FOREIGN KEY (`IDperioada`)
  REFERENCES `rentbike`.`Perioada` (`idPerioada`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- hack pentru

-- ERROR 1452: Cannot add or update a child row: a foreign key constraint fails


SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;


#POPULARE TABELE


INSERT INTO tip (nume)

VALUES

('mountain bike'),

('road bike'),

('children bike'),

('city bike');


INSERT INTO caracteristici(culoare,nr_viteze,tip_frana,diametru_roata,iDtip)

VALUES

('negru', 21,'Mecanica pe disc', 27.5, 1), -- 1
```

('violet', 24,'Mecanica pe disc', 27.5, 1),

('multicolor', 24,'Mecanica pe disc', 26, 1),

('roz', 24,'Mecanica pe disc', 27.5, 1),

('maro', 21,'Mecanica pe disc', 26 , 1),

('alb', 22,'U-brake', 28, 2), -- 6

('alb', 4,'V-brake', 28 , 2),

('negru', 22,'U-brake', 28, 2),

('maro', 2,'Torpedo', 15, 3), -- 9

('crem', 3,'V-brake&Coaster', 20, 3),

('rosu', 3,'V-brake&Coaster', 18.5, 3),

('roz', 0,'Torpedo', 12, 3),

('rosu', 3,'V-brake', 26, 4), -- 13

('maro', 6,'V-Brake Alhonga', 28, 4),

('rosu', 1,'Fata V-brake', 28, 4);

INSERT INTO bicicleta (Producator, Model, IDcaracteristici, TARIF) VALUES

('Giant', 'ATX 27.5', 1, 20),

('DHS', 'Citaddine 2630',13 , 15),

('Pegas', 'Classic 3s', 6, 13),

('Merinda', 'Big Seven 100',2, 15),

('DHS', 'Conture 2863', 7, 23),

('Pegas', 'Terrana 2645', 3, 14),

('Giant', 'Defi 23', 4, 17),

('Merinda', 'Matts J24 ',9 , 17),

('DHS', 'Kreativ 2441', 10, 12),

('DHS', 'Rocket 2041', 11, 12),

('Merinde', 'Minnie Mouse', 12, 15),

('Pegas', 'Magistral', 14, 20),

('Giant', 'SUBWAY 2', 15, 15),

('Giant', 'LIV GIANT AVAIL 1', 8,21),

('Merinda', 'Big Nine 800 M', 5, 17);

INSERT INTO adresa (Oras, Strada, Numar, Cod_postal) VALUES

('Cluj', 'Observatorului' , 32, 21743),

('Suceava', 'Gh. Doja', 10, 23000),

('Alba Iulia', 'Mihai Viteazu', 20, 34001),

('Bistrita', 'Tudoras Gavril', 23, 2400),

('Cluj', 'Dornelor' , 34, 21743),

('Suceava', 'Burdujeni', 15, 23000),

('Alba Iulia', 'Grigore Manolescu', 22, 34001),

('Bistrita', 'Pasteur', 18, 2400),

('Alba Iulia', 'Vasile Sincai', 20, 34001),

('Bistrita', 'Stefan cel Mare', 53, 2400),

('Cluj', 'Unirii' , 94, 21743),

('Suceava', 'Burdujeni', 85, 23000),

('Cluj', 'Avram Iancu' , 84, 21753),

('Cluj', 'Traian' , 12, 21033),

('Cluj', 'Decebal' , 52, 21043);

INSERT INTO cont_bancar (Cont, Data_expirare, Cod_siguranta, Tip) VALUES

(758046415061, '2025-12-23', 193, 'Visa'),

(551454838132, '2023-11-03', 234, 'Visa'),

(758616081215, '2024-02-12', 241, 'MasterCard'),

(510301538402, '2027-05-23', 123, 'MasterCard'),

(534829582595, '2025-04-12', 776, 'Maestro'),

(202030304995, '2023-07-13', 990, 'MasterCard'),

(312532523525, '2022-09-23', 554, 'Visa'),

(475028394021, '2021-03-19', 323, 'MasterCard'),

(912348024850, '2023-06-20', 324, 'Maestro'),

(123144124155, '2025-05-30', 990, 'Visa');

INSERT INTO date_logare(username,parola)

VALUES

('ale', 'ale123'),

('ralu', 'ralu1'),

('dragos', 'dragos0'),

('octav', 'octav1'),

('sergiu', 'sergiu123'),

('ana', 'ana1'),

('paula', 'paula2'),

('sabrina', 'sabrina1'),

('claudia', 'claudia0'),

('ioana', 'ioana0');

INSERT INTO client (CNP, Nume, Prenume, Telefon, Email, IDdate_logare, IDadresa, Cont_Bancar)
VALUES

(1960302033133, 'Varga', 'Alexandra', 0758895249, 'alexandra@gmail.com',1, 1, 758046415061),

(1930102340323, 'Bozdog', 'Raluca', 0748284955, 'raluca@yahoo.com', 2, 2, 551454838132),

(2903245738123,'Lazea', 'Dragos', 0732458384, 'dragos@gmail.com', 3, 3, 758616081215),

(2902345235434,'Stanciu','Octavian', 0754344252, 'octav@gmail.com', 4, 4, 510301538402),

(1934950622213,'Pop','Sergiu', 0753455535, 'sergiu@gmail.com',5, 5, 534829582595),

(1890405232542,'Vultur', 'Ana-Maria', 0765663644, 'ana@gmail.com', 6, 6, 202030304995),

(2991213345556,'Sirghi','Paula', 0767238853, 'paula@yahoo.com',7, 7, 312532523525),

(1960903424144,'Trif', 'Sabrina', 0773424525, 'sabrina@gmail.com', 8, 8, 475028394021),

(2901325355424,'Husaru', 'Claudia', 0756632754, 'claudia@gmail.com', 9, 9,912348024850),

(1970823531234, 'Virna', 'Ioana', 0774125124, 'ioana@yahoo.com', 10, 10, 123144124155);

INSERT INTO locatii(Nume,AdresaID, Ora_Deschidere, Ora_inchidere) VALUES

('Rentbike1',13, 8, 20),

('Rentbike2',14, 7, 18),

('Rentbike3',15, 9, 20);

INSERT INTO discount(IDBicicleta, Pret_nou, Start_oferta, Sfarsit_oferta) VALUES

(1, 17, '2021-2-02', '2021-2-06'),

(3, 10, '2021-1-10', '2021-1-27'),

(6, 11, '2021-1-20', '2021-1-25'),

(7, 14, '2021-2-01', '2021-2-05');

INSERT INTO perioada (Data, Ora_inceput, Ora_sfarsit) VALUES

('2021-02-19', 9, 12),

('2021-02-24', 10, 14),

('2021-03-02', 9, 11),

('2021-03-12', 14, 17),

('2021-03-30', 13, 20),

('2021-04-02', 8, 11),

('2021-04-02', 12, 15),

('2021-04-12', 10, 12),

('2021-04-15', 15, 18),

('2021-04-16', 12, 17),

('2021-04-17', 11, 13),

('2021-05-01', 10, 16),

('2021-05-02', 12, 15);

INSERT INTO rent(IDBicicleta, CNP_client, IDperioada, IDlocatie, Data_rezervarii,pret_total)
VALUES

(5, 1934950622213, 3, 1, '2021-03-01',20),

(8, 2901325355424, 1, 1, '2021-02-17',15),

(3, 2903245738123, 6, 3, '2021-04-01',18),

(1, 1890405232542, 5, 2, '2021-03-27',50),

(5, 2991213345556, 7, 3, '2021-04-01',80),

(6, 1960903424144, 2, 1, '2021-02-20',12),

(2, 1960302033133, 4, 2, '2021-03-09',45),

(4, 2991213345556, 8, 1, '2021-04-10',67),

(13, 2902345235434, 10, 2, '2021-04-13',50),

(10, 1960302033133, 11, 3, '2021-04-17',78),

(1, 1930102340323, 12, 1, '2021-04-30',13),

(9, 1970823531234, 13, 3, '2021-05-01',10),

(5, 1960302033133, 3,2,'2021-01-01',50);

#PROCEDURI STOCATE

#ADAUGARE BICICLETA NOUA IN BAZA DE DATE

#ADAUGARE CLIENT NOU IN BAZA DE DATE

#ANULARE REZERVARE A UNEI BICICLETE DIN BAZA DE DATE

#CREARE REZERVARE A UNEI BICICLETE DIN BAZA DE DATE

#DISPONIBILITATE INCHIRIERE BICICLETA DIN BAZA DE DATE

#SCHIMBARE PAROLA A UNUI CONT DE USER DIN BAZA DE DATE

USE `rentbike`;

-- procedura cu care admin-ul adauga biciclete

```
-- DROP procedure IF EXISTS `ADAUGARE_BICICLETA`;


DELIMITER $$

USE `rentbike`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `ADAUGARE_BICICLETA`(producator varchar(20), model varchar(45), tarif int, culoare varchar(15), nr_viteze int, tip_frana varchar(25), diametru_roata int, tip varchar(20))

BEGIN

        #verific daca nu cumva exista deja bicicleta

        SET @exista = null;

    SELECT @exista := bicicleta.Model from bicicleta where bicicleta.Model = model and bicicleta.Producator = producator;

    #daca nu exista inserez una noua

    IF @exista IS NULL THEN

        #gasesc id-ul tipului de bicicleta in functie de numele tip-ul de bicicleta data de utilizator

        set @idTip = null;

        select @idTip := tip.idTip from tip where tip.Nume = tip;

        INSERT INTO caracteristici(Culoare, Nr_viteze, Tip_frana, Diametru_roata, IDtip) VALUES

        (culoare, nr_viteze, tip_frana, diametru_roata, @idTip);

        #inserez in tabela de caracteristici o noua tupla de caracteristici
```

```
    set @idCarac = null;

    SELECT @idCarac := max(caracteristici.idCaracteristici) from caracteristici; -- are oricum autoincrement, max-ul e tupla null

    INSERT INTO bicicleta (Producator, Model, TARIF, IDcaracteristici) VALUES

    (producator, model, tarif, @idCarac);

      ELSE  SELECT CONCAT("Aceasta bicicleta exista") as rezultat;

      END IF;

END$$


DELIMITER ;


-- procedura cu care putem inregistra un nou client


DROP procedure IF EXISTS `ADUGARE_CLIENT_NOU`;


DELIMITER $$

USE `rentbike`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `ADUGARE_CLIENT_NOU`( CNP bigint, Nume varchar(20), Prenume varchar(20), Telefon int(11), email varchar(30), username varchar(20), parola varchar(25),

 oras varchar(20),strada varchar(20),numar int,cod_postal int,cont bigint(20), data_expirare date, cod_siguranta int, tip varchar(20))
```

```
BEGIN
        #verificam daca nu exista deja clientul
    SET @exista=NULL;
    SELECT @exista :=client.CNP FROM client WHERE client.CNP=CNP;
    #daca nu exista inseram datele clientului nou
    IF @exista IS NULL THEN
                #verificam sa nu existe deja username-ul ales
        SET @exista_username=NULL;
        SELECT @exista_username := date_logare.Username FROM date_logare WHERE
username=date_logare.username;
        #daca username-ul nu exista inseram datele clientului
        IF @exista_username IS NULL THEN
                        #inserare adresa
                        INSERT INTO adresa (Oras, Strada, Numar, Cod_postal) VALUES -- adaug
camp nou
        (oras,strada,numar,cod_postal);
        #idadresa nou
        SELECT @id_adresa:= MAX(adresa.idadresa) FROM adresa; -- ultimul adaugat
        #inserare date logare
        INSERT INTO date_logare(username,parola) VALUES -- adaug camp nou
        (username, parola);
```

```
        #idDateLogare nou

        SELECT @id_logare:= MAX(date_logare.iddate_logare) FROM date_logare; -- ultimul adaugat

        #inserare cont bancar

        INSERT INTO cont_bancar (Cont, Data_expirare, Cod_siguranta, Tip) VALUES

        (cont, data_expirare , cod_siguranta, tip);

        #inserare date client + id-uri campuri noi date_logare, adresa, cont_bancar - completez

        INSERT INTO client (CNP, Nume, Prenume, Telefon, Email, IDdate_logare, IDadresa, Cont_Bancar) VALUES

        (CNP, Nume , Prenume , Telefon , email ,@id_logare,@id_adresa, cont);



                ELSE

      SELECT CONCAT('Exista deja un utilizator cu username-ul ',username) rezultat;

      END IF;

    ELSE

    SELECT CONCAT('Exista deja un utilizator cu CNP-ul ',CNP) rezultat;

    END IF;

END$$



DELIMITER ;
```

```
-- anulare rezervare a unei biciclete

-- DROP procedure IF EXISTS `ANULARE_REZERVARE`;



DELIMITER $$

USE `rentbike`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `ANULARE_REZERVARE`(username varchar(20), model_b varchar(45), producator_b varchar(20), data date, ora_inceput int, ora_sfarsit int)

BEGIN

        SET @exista_client=NULL;

    #verificare daca clientul este inregistrat cu cont in magazinul online si are rezervarea cautata in tabela de rezervari?

    SELECT @exista_client:=r.idRent FROM rent r

                                        JOIN client c ON r.CNP_client=c.CNP

                                        JOIN date_logare dl ON c.IDdate_logare=dl.idDate_logare

                JOIN perioada p ON r.IDperioada=p.idPerioada

                JOIN bicicleta b on r.IDBicicleta = b.idBicicleta

                WHERE username=dl.Username and b.Model = model_b and b.Producator = producator_b

                and p.data=data and p.Ora_inceput=Ora_inceput and p.Ora_sfarsit=Ora_sfarsit;

        #daca are cont in magazin si rezervare corespunzatoare celei introduse => ii anulam rezervarea

    IF @exista_client IS NOT NULL THEN
```

```
                    #caut data in care a facut rezervarea

        SET     @idperioada=NULL;

                    SELECT @idperioada:=perioada.idPerioada FROM perioada where
perioada.data=data and perioada.Ora_inceput=Ora_inceput and perioada.ora_sfarsit=ora_sfarsit;

                    #sterg rezervarea din rent si din perioada => rezervarea nu mai exista

        DELETE FROM rent  WHERE rent.idrent=@exista_client;

        DELETE FROM perioada WHERE idperioada=@idperioada;

        ELSE

            SELECT CONCAT('Nu exista rezervarea!',");-- nu exista rezervarea pe care vrea sa o
anuleze user-ul

        END IF;

    END$$



DELIMITER ;



-- verificare de catre client daca poate inchiria o bicicleta



-- DROP procedure IF EXISTS `DISPONIBILITATE_BICICLETA`;



DELIMITER $$

USE `rentbike`$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `DISPONIBILITATE_BICICLETA`(model varchar(45), data1 date, ora_inceput int, ora_final int, out ok int)

BEGIN



	set @idPer = null;

    set @idrent = null;

    set @idModel = null;



    #caut modelul de bicicleta introdus de client

    SELECT @IdModel := bicicleta.idBicicleta from bicicleta where bicicleta.Model = model;

	-- select @idmodel;

    #caut intervalul de timp in care vrea sa rezerve bicicleta

    SELECT @IdPer := perioada.idPerioada from perioada where (perioada.Data = data1

    and ora_final <= perioada.Ora_sfarsit

	and ora_inceput >= perioada.Ora_inceput);

    #caut in rezervari daca exista o rezervare echivalenta cu perioada si bicicleta introdusa

    SELECT	@idrent := rent.idRent from rent where idPerioada = @IDPer and idBicicleta = @IdModel;

    #daca nu gasim o inregistrare care sa corespunda datelor introduse

    IF @idrent IS NULL THEN
```

```
                -- SELECT CONCAT('Bicicleta disponibila!'); -- inseamna ca bicicleta poate fi
rezervata

        set ok=1;



    ELSE

                -- SELECT CONCAT('Bicicleta indisponibila!');

        set ok=-1;



    END IF;

    END$$



DELIMITER ;



-- creare rezervare a unei biciclete de catre clienti



-- DROP procedure IF EXISTS `CREARE_REZERVARE`;



DELIMITER $$

USE `rentbike`$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `CREARE_REZERVARE`(username VARCHAR(20), model_b varchar(45), producator_b varchar(20), data_rez date, ora_inc int, ora_final int, nume_locatie varchar(20))

BEGIN

	#verificare daca exista user

	set @existaUser=null;

   select @existaUser:= iddate_logare from date_logare where date_logare.username=username;

   #verificare daca exista rezervare pt bicicleta si data dorite ale user-ului

	set @exista = null;

   SELECT @exista := rent.idRent FROM rent

   JOIN client cl on rent.CNP_client = cl.CNP

   JOIN bicicleta b on rent.IDBicicleta = b.idBicicleta

   JOIN perioada p on rent.IDperioada = p.idPerioada

   JOIN date_logare dl ON cl.IDdate_logare = dl.idDate_logare

   WHERE dl.Username = username and b.Model = model_b and b.Producator = producator_b and p.Data = data_rez and p.Ora_inceput = ora_inc and p.Ora_sfarsit = ora_final;

   #nu fac rezervari utilizatorii fara cont de user in baza de date

   IF @existaUser is null THEN

		select concat("Acest user nu exista in baza de date");

	else

   #daca nu exista DEJA rezervarea pe care vrem sa o facem in baza de date => o putem introduce ca rezervare noua
```

```
IF @exista IS NULL THEN

        #introducem intervalul in care dorim sa facem rezervarea

INSERT INTO perioada (Data, ora_inceput , ora_sfarsit) VALUES

(data_rez, ora_inc, ora_final);

set @lastIdPerioada = null;

SELECT @lastIdPerioada := max(idPerioada) FROM perioada as rezultat; -- ultima tupla introdusa

#id date_logare client care face rezervarea

set @idDate_logare = NULL;

SELECT @idDate_logare := date_logare.idDate_logare FROM date_logare WHERE date_logare.Username = username;

#cnp client client care face rezervarea

set @cnp_client = null;

SELECT @CNP_client := client.CNP FROM client WHERE client.IDdate_logare = @idDate_logare;

#id bicicleta rezervata

set @idBicicleta = NULL;

SELECT @idBicicleta := bicicleta.idBicicleta from bicicleta where bicicleta.Model = model_b and bicicleta.Producator = producator_b;

#id locatie de unde rezerv bicicleta

Set @idLocatie = null;

SELECT @idLocatie := locatii.idlocatii from locatii where locatii.Nume = nume_locatie;
```

```
# calculam pret total pentru inchiriere bicicleta in functie de durata inchirierii

set @pret=null;

set @nrOre=null;

set @nrOre=abs(ora_final-ora_inc);

select @pret:= @nrOre * (select bicicleta.TARIF from bicicleta where  bicicleta.Model =
model_b and bicicleta.Producator = producator_b);

#inseram in tabela rent de inchirieri noua rezervare + data curenta in care s-a efectuat rezervarea

INSERT INTO rent (IDBicicleta, CNP_client, IDperioada, IDlocatie, Data_rezervarii,pret_total)
values

(@idBicicleta, @CNP_client, @lastIdPerioada, @idLocatie, current_date(),@pret);



    ELSE SELECT CONCAT("Acest user a mai facut o rezervare pt aceeasi bicicleta in aceeasi
perioada!") as rezultat;



    END IF;

  end if;

END$$


DELIMITER ;


-- procedura de schimbare a parolei cont de user
```

```
-- DROP procedure IF EXISTS `SCHIMBARE_PAROLA`;


DELIMITER $$

USE `rentbike`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `SCHIMBARE_PAROLA`(username varchar(20), parola varchar(25),parola_noua varchar(20))

BEGIN

		#verificam daca exista user-ul in baza de date a magazinului

		SET @exista_username=NULL;

	SELECT @exista_username:=date_logare.idDate_logare FROM date_logare WHERE username=date_logare.username and

	parola=date_logare.Parola;

	#daca exista user-ul putem schimba parola atribuita contului sau

	IF @exista_username IS NOT NULL THEN

	UPDATE date_logare SET date_logare.parola=parola_noua WHERE date_logare.idDAte_logare=@exista_username;

	else

	SELECT CONCAT('Username sau parola incorecta!',");

	END IF;

END$$
```

```
DELIMITER ;


#TRIGGERE TABELE


#CAND UTILIZATORUL FACE REZERVARI VERIFIC CA DATELE CE VREA SA LE INTRODUCA SA NU FIE ERONATE


#1. REZERVARILE SA NU FIE FACUTE INAINTE DE DATA CURENTA DIN CALENDAR

#2. INTERVALUL ORAR PENTRU INCHIRIEREA BICICLETELOR SA NU FIE INVERSAT


-- Daca data in care se face rezervarea este mai mica decat data curenta atunci data rezervarii este data curenta


-- DROP TRIGGER IF EXISTS rentbike.data_AFTER_INSERT;

DELIMITER $$

USE `rentbike`$$

CREATE DEFINER = CURRENT_USER TRIGGER rentbike.data_AFTER_INSERT before insert ON perioada FOR EACH ROW

BEGIN

        IF new.Data < CURDATE() THEN
```

```
            SET new.Data = CURDATE();

        END IF;

END$$

DELIMITER ;


-- Verifica validate interval orar introdus - daca e gresit il inversam


-- DROP TRIGGER IF EXISTS rentbike.orar_AFTER_INSERT;

DELIMITER $$

USE `rentbike`$$

CREATE DEFINER = CURRENT_USER TRIGGER rentbike.orar_AFTER_INSERT before INSERT
ON perioada FOR EACH ROW

BEGIN

        IF new.Ora_inceput > new.Ora_sfarsit then

            set @aux1=new.ora_inceput;

      set @aux2=new.ora_sfarsit;

            set new.Ora_inceput = @aux2;

            set new.Ora_sfarsit = @aux1;

        END IF;

END$$

DELIMITER ;
```

#VEDERI TABELE

#1. Istoric comenzi

-- drop view if exists istoric_comenzi;

CREATE VIEW istoric_comenzi As

SELECT concat(Producator," ", Model) as bicicleta, abs(p.Ora_sfarsit - p.Ora_inceput) as nr_ore, p.Data as data, (abs(p.Ora_sfarsit - p.Ora_inceput)*b.TARIF) as pret_total

FROM rent r

JOIN client c on r.CNP_client = c.CNP

JOIN date_logare dl on c.IDdate_logare = dl.idDate_logare

JOIN perioada p on r.IDperioada = p.idPerioada

JOIN bicicleta b on r.idBicicleta = b.idBicicleta;

SELECT * FROM istoric_comenzi;

#2. Oferta biciclete

-- drop view if exists oferte_biciclete;

CREATE VIEW oferte_biciclete as

SELECT model, Pret_nou , b.TARIF as Pret_vechi, Start_oferta, Sfarsit_oferta

from discount d

```
join bicicleta b on d.idDiscount = b.idBicicleta;

SELECT * FROM oferte_biciclete;



#3 Lista clienti

-- drop view if exists lista_clienti;



CREATE VIEW lista_clienti as

SELECT distinct Nume, Prenume

FROM client;

SELECT * FROM lista_clienti;



#4 Lista clienti plus bicicletele inchiriate

-- drop view if exists client_bic;



CREATE VIEW client_bic as

SELECT nume,prenume,CNP_client,(p.Ora_sfarsit - p.Ora_inceput) ore,model

FROM rent r

JOIN perioada p ON r.IDperioada=p.idPerioada

JOIN client c ON r.CNP_client=c.CNP

JOIN bicicleta b ON r.IDBicicleta=b.idBicicleta
```

```
ORDER BY ore DESC;

SELECT * FROM client_bic;


#INTEROGARI TABELE

-- 1.Afisarea clientilor ce au inchiriat biciclete de pe Stada Avram Iancu

SELECT distinct c.nume,prenume,telefon

FROM client c

JOIN rent r ON c.CNP=r.CNP_client

JOIN locatii l ON r.IDlocatie=l.idlocatii

JOIN adresa a ON l.AdresaId=a.idAdresa

WHERE a.strada='Avram Iancu';


-- 2.Afisarea clientilor ce nu au inchiriat bicilete Giants sau Pegas

SELECT distinct nume,prenume,email

FROM client c

JOIN rent r ON c.CNP=r.CNP_client

JOIN bicicleta b ON r.IDBicicleta=b.idBicicleta

WHERE producator NOT IN( 'Giant','Pegas');


-- 3.Afisarea nume,prenume,username Top clienti
```

```
SELECT nume,prenume,username,t.nr_inchirieri

FROM client c

JOIN date_logare dl on c.IDdate_logare=dl.idDate_logare

JOIN (SELECT CNP_client,COUNT(CNP_client) nr_inchirieri

FROM rent

GROUP BY CNP_CLIENT) t

ON t.CNP_client=c.CNP

ORDER BY t.nr_inchirieri desc,nume,prenume;



-- 4.Afisarea bicicletelor care au un pret mai mic de 15 lei pe ora,fara a avea discount

SELECT Producator,model,tarif

FROM bicicleta b

WHERE  TARIF<=15 and  NOT EXISTS

(SELECT IDBicicleta

FROM discount d

WHERE b.idBicicleta=d.IDBicicleta);



-- 5.durata perioadei in care fiecare client a inchirat biciclete

SELECT nume,prenume,CNP_client,(p.Ora_sfarsit - p.Ora_inceput) ore,model

FROM rent r
```

```
JOIN perioada p ON r.IDperioada=p.idPerioada

JOIN client c ON r.CNP_client=c.CNP

JOIN bicicleta b ON r.IDBicicleta=b.idBicicleta

ORDER BY ore DESC;



-- 6.Nr de biciclete pe care il are DHS grupat dupa tip

SELECT nume,Count(Producator)

FROM bicicleta b

JOIN caracteristici c ON b.IDcaracteristici=c.idCaracteristici

JOIN tip t ON c.IDtip=t.idTip

WHERE Producator='DHS'

GROUP BY nume;



-- 7. clientii si contul bancar care si-au facut rezervari utilizand card de tipul MasterCard

SELECT c.nume, c.prenume,cb.cont

FROM client c

JOIN cont_bancar cb ON c.Cont_Bancar=cb.Cont

JOIN rent r ON c.CNP=r.CNP_client

where tip='MasterCard';
```

```
-- 8. Afisare clienti + oras + user si parola

select cnp, nume, prenume, telefon, email, a.Oras, dl.Username, dl.Parola

from client

join date_logare dl, adresa a

where client.IDdate_logare = dl.idDate_logare and client.IDadresa = a.idAdresa;



-- 9. Clientii care au facut rezervare in luna martie

SELECT c.nume, c.prenume, r.Data_rezervarii

FROM rent r

JOIN client c

WHERE r.CNP_client = c.CNP and Data_rezervarii  BETWEEN '2021-03-01' AND '2021-03-31';



-- 10. Clientii care au inchiriat biciclete de la fima GIANT

SELECT c.nume, c.Prenume,  b.Producator

FROM client c

JOIN rent r, bicicleta b

WHERE r.CNP_client = c.CNP and r.IDBicicleta = b.idBicicleta and b.Producator = 'Giant';



-- 11. Cati oameni au inchiriat fiecare tip de bicicleta

SELECT t.nume, COUNT(r.IDBicicleta) as nr_biciclete_inchiriate
```

```
FROM rent r

JOIN bicicleta b, caracteristici c, tip t

WHERE r.IDBicicleta = b.idBicicleta and b.IDcaracteristici = c.idCaracteristici and c.idTip = t.idTip

group by t.nume;


-- 12. Fiecare client ce tip de bicicleta a inchiriat

SELECT c.nume as nume, r.Data_rezervarii as data_rezervarii, t.Nume as Tip

FROM rent r

JOIN client c ON r.CNP_client = c.CNP

JOIN bicicleta b ON r.IDBicicleta = b.idBicicleta

JOIN caracteristici ca ON b.IDcaracteristici = ca.idCaracteristici

JOIN tip t ON ca.IDtip = t.idTip

order by c.nume asc;


-- 13. Care tipuri de biciclete au fost inchiriate de mai mult de 3 ori

SELECT t.nume, COUNT(r.IDBicicleta) as nr_biciclete_inchiriate

FROM rent r

JOIN bicicleta b, caracteristici c, tip t

WHERE r.IDBicicleta = b.idBicicleta and b.IDcaracteristici = c.idCaracteristici and c.idTip = t.idTip

group by t.nume
```

```
HAVING COUNT(t.nume) >= 3;


-- 14. Clientii care au inchiriat biciclete pt mai mult de 3 ore de al producatorul Giant

SELECT nume, prenume, (p.Ora_sfarsit - p.Ora_inceput) as nr_ore, Producator

FROM rent r

JOIN client c on r.CNP_client = c.CNP

JOIN perioada p ON r.IDperioada = p.idPerioada

JOIN bicicleta b ON r.IDBicicleta = b.idBicicleta

WHERE (p.Ora_sfarsit - p.Ora_inceput) > 3 and b.Producator = 'Giant';


-- 15. Afisez producatorii care au tarife care sunt mai mari decat media

SELECT b1.Producator, b1.model, b1.tarif

FROM bicicleta b1

JOIN bicicleta b2 on b1.idBicicleta = b2.idBicicleta

WHERE b1.TARIF > (SELECT SUM(tarif)/ COUNT(idBicicleta) as medie

FROM Bicicleta);



#UTILIZATORI BAZA DE DATE

-- DROP USER IF EXISTS 'anonim'@'localhost';
```

```
Create user 'anonim'@'localhost' identified by 'anonim';

Grant select  on rentbike.discount to 'anonim'@'localhost';

Grant select  on rentbike.bicicleta to 'anonim'@'localhost';

Grant select  on rentbike.caracteristici to 'anonim'@'localhost';

Grant select  on rentbike.tip to 'anonim'@'localhost';

Grant select  on rentbike.client to 'anonim'@'localhost'; -- revocare dupa modificare

Grant select  on rentbike.date_logare to 'anonim'@'localhost';

Grant insert  on rentbike.client to 'anonim'@'localhost';

Grant insert  on rentbike.date_logare to 'anonim'@'localhost';

Grant insert,select on rentbike.adresa to 'anonim'@'localhost';

Grant insert,select  on rentbike.cont_bancar to 'anonim'@'localhost';

Grant select  on rentbike.locatii to 'anonim'@'localhost';


-- DROP USER IF EXISTS 'client'@'localhost';


Create user 'client'@'localhost' identified by 'client';

Grant select,insert,update  on rentbike.client to 'client'@'localhost';

Grant select,insert,update  on rentbike.date_logare to 'client'@'localhost';

Grant select,insert,update  on rentbike.adresa to 'client'@'localhost';
```

```
Grant select,insert,update  on rentbike.cont_bancar to 'client'@'localhost';

Grant select,insert  on rentbike.perioada to 'client'@'localhost';

Grant select,insert  on rentbike.rent to 'client'@'localhost';

Grant select  on rentbike.bicicleta to 'client'@'localhost';

Grant select  on rentbike.caracteristici to 'client'@'localhost';

Grant select  on rentbike.tip to 'client'@'localhost';

Grant select  on rentbike.discount to 'client'@'localhost';

Grant select  on rentbike.locatii to 'client'@'localhost';


SHOW GRANTS FOR 'client'@'localhost'
```