# **Anexa 1 – Lista de figuri**

|  |  |
| --- | --- |
| **Figura 01.** | Captură de ecran cu un fișier de tip READ.me |
| **Figura 02.** | Modificare tip utilizator direct în baza de date |
| **Figura 03**. | Modificare necesare in „driver-ul” bazei de date |
| **Figura 04**. | Tabelele in baza de date de tip Postgresql |
| **Figura 05**. | Structura de fișiere a aplicației WebGUI |
| **Figura 06.** | Schema a arhitecturii generală ale proiectului |
| **Figura 07**. | Apel către API din WebGUI |
| **Figura 08**. | Răspuns de la API la apelul de mai sus |
| **Figura 09**. | Apel către API din DesktopGUI |
| **Figura 10**. | Răspuns de la API la apelul de mai sus |
| **Figura 11**. | Schema cazurilor de utilizare sistem desktop (DesktopGUI) |
| **Figura 12**. | Schema cazurilor de utilizare sistem web (WebGUI) |
| **Figura 13**. | Diagrama de interacțiune a cererilor din întregul sistem |
| **Figura 14**. | Diagrama de activitate a angajatului de la ghișeu |
| **Figura 15**. | Diagrama de activitate a administratorului |
| **Figura 16** | Schema bazei de date |
| **Figura 17**. | Output consolă conturi |
| **Figura 18**. | Output consolă clienți |
| **Figura 19**. | Output consolă abonamente |
| **Figura 20**. | Output consolă tipuri de abonamente |
| **Figura 21**. | Output consolă SIM-uri |
| **Figura 22**. | Output consola extra opțiuni |
| **Figura 23**. | Output consolă prețuri extra opțiuni |

# **Anexa 2 – Lista de acronime**

|  |  |
| --- | --- |
| *API* | Application Programming Interface (interfața programabilă a aplicației) |
| *WebGUI* | Aplicația web de interfață grafică cu utilizatorul |
| *DesktopGUI* | Aplicația desktop de interfață grafică cu utilizatorul |
| *DB* | Data Base (baza de date) |
| *SQL* | Structured query language (limbaj structurat de interogări) |
| *HTML* | Hyper Text Markup Language |
| *CSS* | Cascading style sheets |
| *JS* | JavaScript |
| *JSON* | JavaScript Object Notation |
| *GUI* | Graphical User Interface (interfață grafică cu utilizatorul) |
| *HTTPS* | Hypertext Transfer Protocol Secure |
| *HTTP* | Hypertext Transfer Protocol |
| *TCP* | Transmission Control Protocol |
| *SIM* | Subscriber Identity Module |
| *ID* | IDentifier (identificator) |
| *UUID* | Universal Unique IDentifier (identificator universal unic) |

# **Anexa 3 – Cod sursă al aplicației**

const hash\_sha256=(message)=>{

return CryptoJS.SHA256 (CryptoJS.enc.Hex.parse(message)).toString();

}

Cod sursă 1. Cod sha256.

//account table structure

const Sequelize = require("sequelize");

const sequelize = require('../DB/DB\_driver');

const { DataTypes } = require('sequelize');

const Account = sequelize.define('account',{

idAccount: {

type: DataTypes.UUID,

defaultValue: DataTypes.UUIDV4,

primaryKey: true,

allowNull:false

},

firstName:{

type: Sequelize.STRING,

allowNull:false,

},

lastName:{

type: Sequelize.STRING,

allowNull:false

},

email:{

type: Sequelize.STRING,

allowNull:false,

},

birthYear:{

type: Sequelize.INTEGER,

allowNull:false,

},

username:{

type: Sequelize.STRING,

allowNull:false,

unique:true

},

password:{

type: Sequelize.STRING,

allowNull:false

},

type:{

type: Sequelize.STRING,

allowNull:false

},

token:{

type: Sequelize.STRING

},

expiery:{

type:DataTypes.DATE

}

});

module.exports = Account;

Cod sursă 2. Cod creare model conturi.

//client table structure

const Sequelize = require("sequelize");

const sequelize = require('../DB/DB\_driver');

const { DataTypes } = require('sequelize');

const Client = sequelize.define('client',{

idClient: {

type: DataTypes.UUID,

defaultValue: DataTypes.UUIDV4,

primaryKey: true,

allowNull:false

},

firstName:{

type: Sequelize.STRING,

allowNull:false,

},

lastName:{

type: Sequelize.STRING,

allowNull:false

},

email:{

type: Sequelize.STRING,

allowNull:false,

},

birthYear:{

type: Sequelize.INTEGER,

allowNull:false,

},

paymentMethod:{

type: Sequelize.ENUM('cash', 'transfer'),

allowNull:false,

},

idAccount: {

type: Sequelize.DataTypes.UUID

}

});

module.exports = Client;

Cod sursă 3. Cod creare model clienți.

//subscription table structure

const Sequelize = require("sequelize");

const sequelize = require('../DB/DB\_driver');

const { DataTypes } = require('sequelize');

const Subscription = sequelize.define('subscription',{

idSubscription: {

type: DataTypes.UUID,

defaultValue: DataTypes.UUIDV4,

primaryKey: true

},

dataStart:{

type: Sequelize.STRING,

allowNull:false

},

reccuring:{

type: DataTypes.STRING,

allowNull:false

},

idSimAsociat: {

type: DataTypes.UUID,

},

idClient: {

type: DataTypes.UUID,

allowNull:false

},

idSubscriptionType: {

type: DataTypes.UUID,

allowNull:false

}

});

module.exports = Subscription;

Cod sursă 4. Cod creare model abonamente.

//subscriptionType table structure

const Sequelize = require("sequelize");

const sequelize = require('../DB/DB\_driver');

const { DataTypes } = require('sequelize');

const SubscriptionType = sequelize.define('subscriptionType',{

idSubscriptionType: {

type: DataTypes.UUID,

defaultValue: DataTypes.UUIDV4,

primaryKey: true

},

nrMessages:{

type: Sequelize.INTEGER,

allowNull:false

},

nrMinutes:{

type: Sequelize.INTEGER,

allowNull:false

},

nrGbInternet:{

type: Sequelize.INTEGER,

allowNull:false

},

price:{

type: Sequelize.INTEGER,

allowNull:false

}

});

module.exports=SubscriptionType;

Cod sursă 5. Cod creare model tipuri de abonamente.

//subscription table structure

const Sequelize = require("sequelize");

const sequelize = require('../DB/DB\_driver');

const { DataTypes } = require('sequelize');

const Sim = sequelize.define('sim',{

idSim: {

type: DataTypes.UUID,

defaultValue: DataTypes.UUIDV4,

primaryKey: true

},

nrSerie:{

type: DataTypes.INTEGER,

allowNull:false

}

});

module.exports = Sim;

Cod sursă 6. Cod creare model *SIM*-uri.

//extraOption table structure

const Sequelize = require("sequelize");

const sequelize = require('../DB/DB\_driver');

const { DataTypes } = require('sequelize');

const ExtraOption = sequelize.define('extraOption',{

idExtraOption: {

type: DataTypes.UUID,

defaultValue: DataTypes.UUIDV4,

primaryKey: true

},

idSubscription: {

type: DataTypes.UUID,

allowNull:false

},

type:{

type: Sequelize.STRING,

allowNull:false

},

number:{

type: Sequelize.INTEGER,

allowNull:false

},

price:{

type: Sequelize.INTEGER,

allowNull:false

}

});

module.exports = ExtraOption;

Cod sursă 7. Cod creare model extra opțiuni.

//extraOptionPricing table structure

const Sequelize = require("sequelize");

const sequelize = require('../DB/DB\_driver');

const { DataTypes } = require('sequelize');

const ExtraOptionPricing = sequelize.define('extraOptionPricing',{

idExtraOptionPricing: {

type: DataTypes.UUID,

defaultValue: DataTypes.UUIDV4,

primaryKey: true

},

type: {

type: DataTypes.STRING,

allowNull:false

},

pricePerUnit:{

type: Sequelize.INTEGER,

allowNull:false

}

});

module.exports = ExtraOptionPricing;

Cod sursă 8. Cod creare model preturi extra opțiuni.