# **JAVA SCRIPT**

**VAR 2-JS 5/5**

**# Subiect 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand urmatoarea functie function `bowdlerize(input, dictionary)` unde:**

- un obiect `Duck` este construit pe baza unui nume de tip string sau String

- un `Duck` poate sa inoate

- un `RubberDuck` poate sa pluteasca

- un `RubberDuck` nu poate sa inoate

**# Completati urmatoarele taskuri:**

- `name` trebuie sa fie de tip `string` sau `String`. Daca este furnizat alt tip este aruncat `Error` cu mesajul`name must be string or String`; (0.5 pts)

- daca i se cere unui obiect Duck sa inoate va returna un string (ex. daca avem un Duck cu numele `Donald` atunci string-ul returnat este `Donald is swimming`); (0.5 pts)

- o rata de cauciuc este atat de tip `Duck` cat si `RubberDuck`; (0.5 pts)

- daca i se cere unui obiect RubberDuck sa pluteasca va returna un string (ex. daca avem un RubberDuck cu numele `Donald` atunci string-ul returnat este `Donald floats`); (0.5 pts)

- daca i se cere unui obiect RubberDuck sa inoate va returna un string (ex. daca avem un RubberDuck cu numele `Donald` atunci string-ul returnat este `Donald can't swim, only float`); (0.5 pts)

class Duck {

    constructor(name) {

//cerinta 1

        if (typeof name === 'string' || typeof name === String) {

            this.name = name

        }

        else throw new Error(`name must be string or String`)

    }

    move() {

        return `${this.name} is moving`

    }

    swim() {

        return `${this.name} is swimming`

    }

}

class RubberDuck extends Duck{

    constructor(name){

        super(name)

    }

    float() {

        return `${this.name} floats`

    }

    swim() {

        return `${this.name} can't swim, only float`

    }

}

const app = {

    Duck,

    RubberDuck

}

module.exports = app

**2) Var 2 Alice 5/5 sau examen Robert**

**# Subiect 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand clasa `Queue` din fisierul `index.js` sa se implementeze urmatoarele functionalitati:**

- clasa `Queue` trebuie sa contina o proprietate numita `items`, de tip `Array` care va fi initializata cu un vector gol (0.5 pts);

- se va implementa metoda `insert` ce va primi ca parametru un `element` care va fi adaugat in vector, conform principiului cozii;

- metoda `insert` va permite adaugarea in coada doar a elementelor de tip `string` si va arunca o eroare cu textul `Invalid Type` pentru elementele de alt tip.

- se va implementa metoda `extract` ce va returna un `element` din vector, conform principiului cozii;

- in cazul in care vectorul este gol si se apeleaza metoda `extract`, se va

arunca o eroare cu textul `Invalid Operation`;

class Queue {

    constructor() {

        this.items = []

    }

    insert = (element) => {

        if (typeof element === 'string') {

            this.items.push(element)

        }

        else throw new Error('Invalid Type')

    }

    extract = () => {

        if (this.items.length > 0) {

            return this.items.shift()

        }

        else throw new Error(`Invalid Operation`)

    }

}

module.exports = Queue;

**3)JS ANA V0 5/5**

**# Subiect 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand urmatoarea functie function `bowdlerize(input, dictionary)` unde:**

- `input` este un string (ex. "This is a cat")

- `dictionary` este un vector ce contine o serie de string-uri.

**# Completati urmatoarele taskuri:**

- `input` trebuie sa fie de tip `string`. Daca este dat un input de alt tip se va arunca un `Error` cu mesajul `Input should be a string`; (0.5 pts)

- `dictionary` este un vector de elemente de tip `string`. Daca cel putin un element nu este `string` se va arunca un `Error` cu mesajul `Invalid dictionary format`; (0.5 pts)

- Daca `dictionary` contine cuvinte, vor fi inlocuite in `input` cu prima litera, urmată de  o serie de caractere `\*` urmate de ultima litera. Lungimea cuvantului rezultat este acceasi ca a celui initial (ex. 'test' va deveni 't\*\*t') ; (0.5 pts)

- Se va returna un nou string, `input` nefiind modificat; (0.5 pts)

- Functia returneaza rezultatul corect si pentru cuvinte care incep cu litere mari. (0.5 pts)

function bowdlerize(input, dictionary){

    if(typeof input!=='string'){

        throw new Error('Input should be a string')

    }

    var inputCopy = input

    dictionary.forEach(element => {

        if (typeof element !== 'string') {

            throw "Invalid dictionary format"

        }

        //prima litera din cuvant toUpperCase

        var element2 = element.charAt(0).toUpperCase() + element.slice(1)

        //daca inputul

        if(inputCopy.includes(element)){

            //despart pe litere

            let letters = element.split('')

            let str = ''

            letters.forEach(letter => {

                //daca litera e pe pozitia de inceput sau pe ultima pozitie, o adaugam la cuvantul modificat

                if (letter.toLowerCase() === letters[0].toLowerCase() || letter.toLowerCase() === letters[letters.length - 1].toLowerCase()) {

                    str += letter

                }

                else {

                    //altfel adaugam \* la cuvantul modificat

                    str += '\*'

                }

            })

            inputCopy = inputCopy.replace(element, str)

        }

        else{

            //daca include cuvant cu majuscula

            if(inputCopy.includes(element2)){

                let letters = element2.split('')

                let str = ''

                letters.forEach(letter => {

                    if (letter.toLowerCase() === letters[0].toLowerCase() || letter.toLowerCase() === letters[letters.length - 1].toLowerCase()) {

                        str += letter

                    }

                    else {

                        str += '\*'

                    }

                })

                inputCopy = inputCopy.replace(element2, str)

            }

        }

    });

    return inputCopy

}

const app = {

    bowdlerize

};

module.exports = app;

**4) JS ANA V1 5/5 sau Andrei V1**

**# Subiect 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand urmatoarea functie function `calculateFrequencies(input, stopWords)` unde:**

- `input` este un string sau un obiect String (ex. "This is a cat")

- `stopWords` este un vector ce contine o serie de string-uri sau obiecte String.

**# Completati urmatoarele taskuri:**

- `input` trebuie sa fie de tip `string` sau `String`. Daca este dat un input de alt tip se va arunca un `Error` cu mesajul `Input should be a string`; (0.5 pts)

- `dictionary` este un vector de elemente de tip `string` sau `String`. Daca cel putin un element nu este `string` se va arunca un `Error` cu mesajul `Invalid dictionary format`; (0.5 pts)

- functia calculeaza frecventele relative ale cuvintelor in input si returneaza un dictionar care are cuvintele drept chei si frecventele drept valori (ex. pentru string-ul 'orange cat' rezultatul va fi {orange : 0.5, cat : 0.5}); (0.5 pts)

- daca stopWords contine cuvinte, ele vor fi ignorate pentru calcularea frecventelor (ex. pentru 'the orange cat' cu 'the' in stopWords rezultatul va fi {orange : 0.5, cat : 0.5}); (0.5 pts)

- functia returneaza rezultatul corect si pentru cuvinte care incep cu o litera mare, care sunt considerate identice cu varianta fara litera mare. (0.5 pts)

function calculateFrequencies(input, dictionary) {

    if (typeof input !== 'string' && typeof input !== 'String') {

        throw "Input should be a string"

    }

    dictionary.forEach(element => {

        if (typeof element !== 'string' && typeof element !== 'String') {

            throw "Invalid dictionary format"

        }

    })

    //despart inputul (text) in cuvinte

    input = input.split(' ')

    dictionary.forEach(word => {

        input.forEach(element => {

            if (element.toLowerCase() === word.toLowerCase()) {

                //sterge elementul din input

                input.splice(input.indexOf(element), 1)

            }

        })

    })

    let obj = {}

    input.forEach(word => {

        if (!obj.hasOwnProperty(word.toLowerCase())) {

            obj[word.toLowerCase()] = 1

        }

        else {

            obj[word.toLowerCase()]++

        }

    })

    for (pr in obj) {

        obj[pr] = obj[pr] / input.length

    }

    return obj

}

const app = {

    calculateFrequencies

};

module.exports = app;

**5)JS Ana V4 5/5**

**# Subiect 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand urmatoarea functie  `removeOrderItem(orderInfo, position)` unde:**

- `orderInfo` este un obiect cu proprietatile total si items

- `position` este un numar intreg ce determina un element din items

**# Completati urmatoarele taskuri:**

- validati proprietatea `items` sa fie de tip `array`. Daca este dat un input de alt tip se va arunca eroare cu mesajul `Items should be an array`; (0.5 pts)

- fiecare obiect din `items` este are proprietatile price si quantity. Daca cel putin un element nu respecta formatul se va arunca mesajul `Malformed item`; (0.5 pts)

- parametrul `position` este valid in relatie cu vectorul items (0.5 pts)

- se va returna obiectul orderInfo din care s-a eliminat elementul de pe pozitia data (0.5 pts)

- functia returneaza totalul corect pentru comanda in urma eliminarii elementului de pe pozitia data. (0.5 pts)

function removeOrderItem(orderInfo, position){

    if(!Array.isArray(orderInfo.items)){

        throw 'Items should be an array';

    }

    orderInfo.items.forEach(element=>{

    if(!(element.hasOwnProperty('price'))||!(element.hasOwnProperty('quantity')))

    {

        throw 'Malformed item';

    }

    });

    if(position>orderInfo.items.length){

        throw 'Invalid position';

    }

    let p=orderInfo.items[position].price;

    let q=orderInfo.items[position].quantity;

    orderInfo.items.splice(position,1);

    orderInfo.total=orderInfo.total-p\*q;

    return orderInfo;

    }

    const app = {

        removeOrderItem

    };

    module.exports = app;

**Stefi JS V0 5/5**

/\*

 - funcția distance primește ca parametrii două array-uri

 - fiecare element poate apărea cel mult o dată într-un array; orice apariții suplimentare sunt ignorate

 - distanța dintre cele 2 array-uri este numărul de elemente diferite dintre ele

 - dacă parametrii nu sunt array-uri se va arunca o excepție ("InvalidType")

\*/

/\*

 - the distance function receives as parameters two arrays

 - each element can appear in each array at most once; any duplicates are ignored

 - the distance between the 2 arrays is the number of different elements between them

 - if the parameters are not arrays an exception is thrown ("InvalidType")

\*/

function distance(first, second){

    if(!(first instanceof Array) || !(second instanceof Array)){

        throw new Error('InvalidType');

    }

    else{

        var duplicates = 0;

        var f = new Set(first);

        var s = new Set(second);

        first = [...f];

        second=[...s];

        for(var i=0;i<first.length;i++){

            for(var j=0; j<second.length;j++){

                if(first[i]===second[j]){

                    duplicates++;

                }

            }

        }

        var dist=(first.length+second.length)-(2\*duplicates);

        if(first.length==0 && second.length==0){

            return 0;

        }

        return dist;

    }

}

module.exports.distance = distance

**Stefi JS V1 5/5**

/\*

Definiți un tip obiectual (Widget)

Funcția decorate adaugă la Widget o metodă numită enhance, care crește mărimea unui widget cu "n"

Dacă parametrul trimis nu este un număr, se aruncă o excepție ("InvalidType")

Metoda funcționează și asupra Widget-urilor deja declarate

\*/

/\*

Define a Widget object type is defined

The decorate function adds to Widget a method called enhance which increases the size of a widget with n

If the parameter is not a number an exception is thrown ("InvalidType")

The method also works on already declared Widgets

\*/

class Widget {

constructor(name,size){

       this.name=name;

       this.size=size;

    }

    // TODO: completați implementarea

    // TODO: finish the implementation

    getDescription(){

        return `a ${this.name} of size ${this.size}`

    }

}

function decorate(){

    Widget.prototype.enhance=function(n){

        if(typeof n !="number")

        {

            throw new Error("InvalidType");

        }

        this.size+=n;

    }

}

module.exports.decorate = decorate

module.exports.Widget = Widget

**Stefi JS V2 5/5**

/\*

Exista un tip obiectual definit (Bird)

Să se definească tipul Penguin.

Un pinguin este un tip copil pentru Bird și are în plus metoda swim(distance)

Un pinguin nu poate fi creat fără un nume de tip string

Un pinguin nu poate să zboare și va spune asta dacă i se cere

Dacă se apelează makeNest, un pinguin va apela metoda părintelui său

Vedeți testele pentru formatul exact al mesajelor

\*/

/\*

There is an object type called Bird

Define the Penguin type

A penguin is a child type for Bird and has an additional method called swim(distance)

A penguin cannot be created without a name of type string

A penguin cannot fly and will say that if asked

A penguin can make a nest via its parent's method

See the tests for the accurate format of messages

\*/

class Bird {

    constructor(name){

        this.name = name

    }

    fly(distance){

        return `${this.name} flies ${this.distance}`

    }

    makeNest(){

        return `${this.name} makes a nest`

    }

}

class Penguin  extends Bird {

    constructor(name){

        if (!name || typeof name !== 'string'){

            throw new Error('CreationError')

        }

    super(name);

    }

    fly(distance){

        return `${this.name} is a penguin and cannot fly`

    }

    swim(distance){

        return `${this.name} swims ${distance}`

    }

}

// TODO: implementați tipul Pinguin

// TODO: implement the Penguin type

module.exports.Bird = Bird

module.exports.Penguin = Penguin

**Stefi JS V3 5/5**

 /\*

 - funcția capitalize primește ca parametrii un string și un array

 - dicționarul conține o serie de termeni

 - in textul ințial cuvintele sunt separate de spațiu

 - fiecare termen din dicționar trebuie să apară capitalizat în rezultat

 - rezultatul este un string nou, fără modificarea celui inițial

 - dacă textul nu este un string sau dicționarul nu este un array de string-uri se va arunca o excepție (mesajul TypeError)

\*/

/\*

 - the capitalize function receives as parameters a string and an array

 - the dictionary (the array) contains a series of words

 - in the initial text the words are separated by space

 - each dictionary term has to appear capitalized in the result

 - the result is a new string without modifying the initial one

 - if the text is not string or the dictionary not an array of strings an exception is thrown (message is TypeError)

\*/

function capitalize(text, dictionary){

  if(typeof text!=="string" || !Array.isArray(dictionary))

  {

    throw new Error("TypeError");

  }

  if (dictionary.filter((e) => typeof e !== 'string').length){

        throw new Error('TypeError')

    }

  let result = text

    let items = result.split(' ')

    for (let i = 0; i < items.length; i++){

        if (dictionary.indexOf(items[i]) !== -1){

            items[i] = items[i][0].toUpperCase() + items[i].slice(1, items[i].length)

        }

    }

    return items.join(' ')

}

module.exports.capitalize = capitalize

**Stefi JS V4 5/5**

/\*

 - funcția translate primește ca parametrii un string și un obiect

 - funcția aruncă excepții dacă tipurile nu sunt respectate (mesajul "InvalidType")

 - obiectul dicționar are în cheie valoarea inițială și în valoare traducerea ei

 - valorile din dicționar sunt string-uri

 - funcția înlocuiește fiecare cheie din dicționar găsită în textul inițial cu valoarea tradusă

\*/

/\*

 - the translate function receives as parameters a string and an object

 - the function throws exceptions if the types are not the required ones (message is "InvalidType")

 - the dictionary object has in its keys the inital values and in its values the translation of the key

 - the values in the dictionary are strings

 - the function replaces each dictinary key found in the initial text with the value in the dictionary corresponding to the key

\*/

function translate(text, dictionary){

    // TODO: implementați funcția

    // TODO: implement the function

    if( typeof text!=="string")

    {

        throw new Error("InvalidType");

    }

    if(typeof dictionary!=="object" || !dictionary)

    {

        throw new Error('InvalidType');

    }

    for (let prop in dictionary){

        if (typeof dictionary[prop] !== 'string'){

            throw new Error('TypeError')

        }

    }

    let result = text.split(' ')

    for (let prop in dictionary){

        let position = result.indexOf(prop)

        if (position !== -1){

            result[position] = dictionary[prop]

        }

    }

    return result.join(' ')

}

module.exports.translate = translate

**Stefi JS V5 5/5**

**# Subiect 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand urmatoarea functie `function textProcessor(input, tokens)` unde:**

- `input` este un string ce poate sa contina tokenuri (Example: "Hello ${user}" or "Hello")

- `tokens` este un vector ce contine obiecte cu numele si valoarea tokenurilor.

- Toate tokenurile sunt idetificate sub urmatorul format: `${tokenName}`

function textProcessor(input, tokens){

    if(typeof input=="string")

    {

        if(input.length>=6)

        {

            tokens.forEach((e)=>

            {

                if( typeof e.tokenName=="string" && typeof e.tokenValue=="string" )

                {

                    var t= "${"+ e.tokenName + "}";

                    input= input.replace(t, e.tokenValue);

                }

                else

                {

                    throw new Error("Invalid array format");

                }

            });

            return input;

        }

        else

        {

        throw new Error("Input should have at least 6 characters");

        }

    }

    else

    {

        throw new Error("Input should be a string");

    }

}

const app = {

    textProcessor: textProcessor

};

module.exports = app;

**Stefi JS V6 – pica**

**# Subiectul 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand urmatoarea functie `function applyBonus(employees, bonus)`, completati urmatoarele taskuri:**

- Functia trebuie sa returneze un Promise; (0.5 pts)

- Daca `bonus` nu este numar, functia trebuie sa apeleze `reject` cu `Error` si mesajul `Invalid bonus`; (0.5 pts)

- `employees` este un vector ce contine elemente cu urmatorul format: `{name: string, salary: number}` (Example: [{name: "John Doe", salary: 5000}]). Daca este pasat un vector cu elemente invalide, functia trebuie sa apeleze `reject` cu `Error` si mesajul `Invalid array format`; (0.5 pts)

- Functia trebuie sa apeleze `reject` cu `string` cu valoarea `Bonus too small` daca `bonus` este mai mic de 10% din salariul maxim din `employees` array; (0.5 pts)

- Functia trebuie sa apeleze `resolve` cu un vector ce contine salariile marite pentru fiecare angajat; (0.5 pts)

**Stefi JS V7 5/5**

**# Subject 2 (2.5 pts)**

**# TOPIC: Javascript**

**# Having the `function applyDiscount(vehicles, discount)`, complete the following tasks:**

- Function should return a Promise; (0.5 pts)

- If `discount` is not a number, the function should `reject` an `Error` with the message `Invalid discount`; (0.5 pts)

- `vehicles` is an array that contains objects with the following format: `{make: string, price: number}` (Example: [{make: "Audi A5", price: 15000}]). If an array with invalid objects is passed then the function should `reject` an `Error` with the message `Invalid array format`; (0.5 pts)

- Function should `reject` a `string` with the value `Discount too big` if `discount` is greater than 50% of the min price from `vehicles` array; (0.5 pts)

- Function should `resolve` an array with applied discount to each `vehicle price`; (0.5 pts)

function applyDiscount(vehicles, discount){

      return new Promise((resolve, reject) => {

          if(typeof discount=="number")

          {var minprice=9999999;

             vehicles.forEach((e)=>

             { if(typeof e.make!="string" && typeof e.price!="number")

                {

                var eroare2="Invalid array format";

                reject(new Error(eroare2));

                 }

                 else

                 {

                     if(e.price<minprice)

                     {

                         minprice=e.price;

                     }

                 }

             });

             if(discount> minprice\*0.5)

             {

                 reject("Discount too big");

             }

              else {resolve(vehicles.map(e=>{

                    return{make: e.make,price:e.price-discount};

                })); }

          }

          else

          {

              var eroare1='Invalid discount';

              reject(new Error(eroare1));

          }

      });

}

const app = {

    applyDiscount: applyDiscount

};

module.exports = app;

**Stefi JS V8 5/5**

**# Subiectul 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand urmatoarea functie `function addTokens(input, tokens)` unde:**

- `input` este un string ce poate sa contina "...". De exemplu: Subsemnatul ..., dominiciliat in ...;

- `tokens` un vector de tokenuri.

- Functia trebuie sa inlocuiasca toate `...` din `input` cu valorile corespunzatoare din `tokens` sub urmatorul format `${tokenName}`, in ordinea in care exista in vector;

**# Complete the following tasks:**

- `input` trebuie sa fie de tip `string`. Daca alt tip este pasat ca si parametru aruncati `Error` cu mesajul `Input should be a string`; (0.5 pts)

- `input` trebuie sa aiba cel putin 6 caractere ca si lungime. Daca dimensiunea `input-ului` este mai mica de 6, aruncati `Error` cu mesajul `Input should have at least 6 characters`; (0.5 pts)

- `tokens` este un vector de elemente cu urmatorul format: `{tokenName: string}`. Daca urmatorul format nu este respectat, aruncati `Error` cu urmatorul mesaj `Invalid array format`; (0.5 pts)

- Daca `input` nu contine `...` returnati valoarea initiala a `input-ului`; (0.5 pts)

- Daca `input` contine `...`, inlocuiti-le cu valorile specifice si returnati noul `input`; (0.5 pts)

function addTokens(input, tokens){

     if( typeof input!=="string"){

         throw new Error("Invalid input");

     }

    else

    {

        if(input.length <6)

        {

            throw new Error("Input should have at least 6 characters");

        }

    else {

        tokens.forEach((e)=>

         {

         if( typeof e.tokenName !=="string")

         {

             throw new Error("Invalid array format");

         }

         else

         {

         }

        })

        var inputnou;

        if(input.includes("..."))

        { tokens.forEach((e)=>{

            inputnou=input.replace("...", "${" + e.tokenName +"}");});

            return inputnou;

        }

        else

        {return input;

        }

    }

    }

}

const app = {

    addTokens: addTokens

}

module.exports = app;

**Stefi JS V9 5/5**

**# Subiect 2 (2.5 pts)**

**# TOPIC: Javascript**

**# Avand urmatoarea functie `function applyBlackFriday(products, discount)` unde:**

- `products` un vector de obiecte cu urmatorul format {name: string, price: number};

- `discount` un numar ce reprezinta discount-ul ce va fi aplicat preturilor produselor.

- Functia trebuie sa returneze un vector cu preturile reduse pentru toate produsele.

**# Completati urmatoarele cerinte:**

- Functia trebuie sa returneze un promise; (0.5 pts)

- `discount` trebuie sa fie un numar, in caz contrar apelati `reject` in promise cu `Error` si mesajul `Invalid discount`; (0.5 pts)

- `discount` trebuie sa fie mai mare ca 0 si mai mic egal cu 10, in caz contrar apelati `reject` in promise cu `Error` si mesajul `Discount not applicable`; (0.5 pts)

- `products` trebuie sa contina produse cu formatul specificat, in caz contrar se va apela `reject` cu `Error` si mesajul `Invalid array format`; (0.5 pts)

Un produs: {name: string, price: number}

- Functia trebuie sa returneze un vector cu preturile reduse pentru toate produsele; (0.5 pts)

function applyBlackFriday(products, discount){

    return new Promise((resolve, reject) => {

        if(typeof discount=="number") {

            products.forEach((e)=>

            {

                if(typeof e.name!="string" && typeof e.price!="number" )

                {  var eroare1='Invalid array format';

                      reject(new  Error(eroare1));

                }

            });

        if(discount<=10 && discount>0)

        {      resolve(products.map(e=>{

                    return{name: e.name,price: e.price- e.price\*discount/100};

                }));

        }

        else

        {

            var  eroare2="Discount not applicable";

             reject(new Error(eroare));

        }

        }

        else

        { var  eroare="Invalid discount";

             reject(new Error(eroare));

        }

     })

}

const app = {

    applyBlackFriday: applyBlackFriday

};

module.exports = app;

**Stefi JS V10 5/5**

**# Subiect 2 (2.5 pts)**

**# TOPIC: Javascript**

**# Dată fiind funcția `function processString(input)`, care inițial tokenizează string-ul `input` în mai multe `tokens` separate de spațiu, rezolvați următoarele cerințe:**

- Dacă oricare `token` nu este un `number` sau un `Number`, funcția ar trebui să arunce `Error`

- Dacă oricare `token` nu este un `number` sau un `Number`, funcția ar trebui să arunce `Error` cu mesajul `Item is not a number`; (0.5 pts)

- Dacă `input` are lungime 0 funcția ar trebui să returneze 100; (0.5 pts)

- Token-urile `impare` sunt ignorate; (0.5 pts)

- Funcția returnează 100 minus suma tuturor `token`-urilor pare; (0.5 pts)

function processString(input){

    if(input.length==0){

        return 100;

    }else{

        let sir=input.split(' ');

        let sum=0;

        for(var i=0;i<sir.length;i++){

            if(!isNaN(sir[i])){

                if(parseInt(sir[i])%2==0){

                    sum+=parseInt(sir[i]);

                }

            }else{

                console.log(typeof parseInt(sir[i]));

                throw new Error("Item is not a number");

            }

        }

        return 100-sum;

    }

}

const app = {

    processString: processString

}

module.exports = app

**Stefi JS V11 5/5**

**# Subiect 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand urmatoarea functie `function getAverageGrade(grades)`, completati urmatoarele taskuri:**

- Funcția aruncă o eroare dacă există o notă negativă (0.5 pts)

- Funcția aruncă o eroare dacă nota nu este fie un număr fie 'A'; (0.5 pts)

- Funcția aruncă o eroare dacă nota nu este definită; (0.5 pts)

- Functia returnează nota medie; (0.5 pts)

- Funcția returnează nota 0 pentru o listă vidă de note (0.5 pts)

function getAverageGrade(gradeItems){

     var sum=0;

    for(let i =0 ;i < gradeItems.length ;i++)

    {

        if(typeof gradeItems[i].grade !=="number"  ||  gradeItems[i].grade=='A' || typeof gradeItems[i].grade ==="undefined" || gradeItems[i].grade <0 )

        {

            throw new Error("Invalid grade");

        }

        else

        {

             sum+=gradeItems[i].grade;

        }

    }

    if(gradeItems.length===0)

    {

        return 0;

    }

    else{

    return  sum/gradeItems.length;

    }

}

const app = {

    getAverageGrade: getAverageGrade

}

module.exports = app

**Stefi JS V11 – 5/5**

**# Subiect 2 (2.5 pts)**

**# Tematica: Javascript**

**# Avand definită clasa `Shape` rezolvați următoarele taskuri:**

- Dacă `Shape` este instanțiată direct și este apelată metoda `area`, se va arunca un obiect `Error`  cu mesajul `not implemented`; (0.5 pts)

- Se va defini clasă `Square` care extinde `Shape`; Un `Square` poate fi instanțiat pe baza unui obiect cu o proprietate `width`; (0.5 pts)

- Dat fiind un `Square`, se va calcula corect aria acestuia; (0.5 pts)

- Se va defini clasă `Circle` care extinde `Shape`; Un `Circle` poate fi instanțiat pe baza unui obiect cu o proprietate `radius`; Dat fiind un `Circle`, se va calcula corect aria acestuia; (0.5 pts)

- Se va defini clasă `Rectangle` care extinde `Shape`; Un `Rectangle` poate fi instanțiat pe baza unui obiect cu o proprietate `width` și o proprietate `height`; Dat fiind un `Rectangle`, se va calcula corect aria acestuia; (0.5 pts)

class Shape{

    constructor(dimensions){

        this.dimensions = dimensions

    }

    area(){

        throw Error('not implemented');

    }

}

// TODO: Square, Circle, Rectangle

class Square extends Shape{

    constructor(dimensions,width)

    {

        super(dimensions);

        var {width:w}=dimensions;

        this.width=w;

    }

    area()

    {

        console.log(this.width);

        return this.width\*this.width;

    }

}

class Circle extends Shape{

    constructor(dimensions,radius)

    {

        super(dimensions);

        var {radius:r}=dimensions;

        this.radius=r;

    }

    area()

    {

        return parseInt(Math.PI\*this.radius\*this.radius);

    }

}

class Rectangle extends Shape{

    constructor(dimensions,width,height)

    {

        super(dimensions);

        var {width:w}=dimensions;

        var {height:h}=dimensions;

        this.width=w;

        this.height=h;

    }

    area()

    {

        return this.width\*this.height;

    }

}

const app = {

  Shape: Shape,

  Square : Square,

  Circle : Circle,

  Rectangle : Rectangle

}

module.exports = app

# **REACT**

**Alice 2 sau Andrei v2**

**# Subiect 4 5/5**

**# Tematica: REACT**

**# Avand urmatoarea aplicatie create folosind `create-react-app`, modificati `Company` si adaugati `CompanyDetails` astfel incat:**

- aplicatia se deseneaza corect (0.5 pts);

- `CompanyDetails` are o proprietate numita `item` care contine compania ale carei detalii le afiseaza (0.5 pts);

- `CompanyList` se deseneaza ca o lista de  `Company`, iar fiecare `Company` are un buton cu eticheta `select` (0.5 pts);

- Daca se da click pe butonul select al unui `Company` se deschide afiseaza componenta de detalii (0.5 pts);

- Daca este afisat `CompanyDetails` si se da click pe butonul cancel se afiseaza din nou lista de companii (0.5 pts);

import React, { Component } from 'react'

class Company extends Component {

  render() {

    let {item} = this.props

    return (

      <div>

        Name {item.name} with {item.employees} employees {item.revenue} revenue

        <button id="select" value="select" onClick={() => this.props.onSelect(item)}>Select</button>

      </div>

    )

  }

}

export default Company

import React, { Component } from 'react'

class CompanyDetails extends Component {

    constructor(props) {

        super(props)

        this.state = {

            item: this.props.item

        }

    }

    render() {

        let item  = this.state.item

        return (

            <div>

                Name {item.name} with {item.employees} employees {item.revenue} revenue

                <button id="cancel" value="cancel" onClick={() => this.props.onCancel()}>Cancel</button>

            </div>

        )

    }

}

export default CompanyDetails

import React, { Component } from 'react'

import CompanyStore from '../stores/CompanyStore'

import Company from './Company'

import CompanyDetails from './CompanyDetails'

class CompanyList extends Component {

    constructor() {

        super()

        this.state = {

            companies: [],

            company: {},

            selected: false

        }

    }

    componentDidMount() {

        this.store = new CompanyStore()

        this.setState({

            companies: this.store.getAll()

        })

        this.store.emitter.addListener('UPDATE', () => {

            this.setState({

                companies: this.store.getAll()

            })

        })

    }

    onSelect = item => {

        this.setState({

            selected: true,

        })

        this.setState({

            company: item

        })

    }

    onCancel = () => {

        this.setState({ selected: false });

    }

    render() {

        if (this.state.selected) {

            return(

                <CompanyDetails item={this.state.company} onCancel={this.onCancel} />

            )

        }

        else {

            return (

                <div>

                    {

                        this.state.companies.map((e, i) =>

                            <Company item={e} key={i} onSelect={this.onSelect} />

                        )

                    }

                </div>

            )

        }

    }

}

export default CompanyList

**2) Var 3 Alice 5/5 sau examen Robert**

**# Subiect 4**

**# Topic: REACT**

**# Avand urmatoarea aplicatie create folosind modulul `create-react-app` completati urmatoarele cerinte:**

- Adaugati componenta `AddDevice` in template-ul componentei `DeviceList`;

- Componenta `AddDevice` trebuie sa contina 2 elemente de tip `input` cu `id`: `name` si `price`;

- Componenta `AddDevice` trebuie sa contina un element de tip `button` cu textul `Submit`, folosit pentru a apela metoda `addItem`;

- Componenta `AddDevice` din interiorul componentei `DeviceList` trebuie sa contina o proprietate numita `onAdd` in obiectul `props`;

- La apasarea butonului `Submit` un nou element va fi afisat si adaugat in starea componentei `DeviceList`;

**### INFORMATII UTILE: Obiectele adaugate in vectorul din starea componentei `DeviceList` sunt de forma { name: String, price: Number }.**

import React from 'react';

class AddDevice extends React.Component {

    constructor(props){

        super(props)

        this.state = {

            name: '',

            price: 0

        }

    }

    handleChangeName = (e) => {

        this.setState({

            name: e.target.value

        })

    }

    handleChangePrice = (e) => {

        this.setState({

            price: e.target.value

        })

    }

    render() {

        return (

            <div>

                <input type="text" id="name" name="name" value={this.state.name} onChange={(e) => {this.handleChangeName(e)}} ></input>

                <input type="text" id="price" name="price" value={this.state.price} onChange={(e) => {this.handleChangePrice(e)}} ></input>

                <button value="submit" onClick={() => {

                    this.props.onAdd(this.state.name, this.state.price)

                }}>submit</button>

            </div>

        )

    }

}

export default AddDevice;

import React from 'react';

import AddDevice from './AddDevice'

class DeviceList extends React.Component {

    constructor() {

        super();

        this.state = {

            devices: []

        };

    }

    add = (name, price) => {

        let p = parseInt(price);

        let dev = this.state.devices;

        dev.push({

            name: name,

            price: p

        })

        this.setState({ device: dev })

    }

    render() {

        return (

            <div>

                <AddDevice onAdd={this.add} />

            </div>

        )

    }

}

export default DeviceList;

**3)React Ana V0 5/5 sau Andrei V0 React**

**# Subiect 4**

**# Tematica: REACT**

**# Avand urmatoarea aplicatie create folosind `create-react-app`, adaugati o componenta `Company` astfel incat:**

- aplicatia se deseneaza corect (0.5 pts);

- `CompanyList` este desenata ca un copil al lui `App` (0.5 pts);

- `CompanyList` se deseneaza ca o lista de  `Company` (0.5 pts);

- `Company` are o proprietate numita `item` care contine compania pe care o afiseaza (0.5 pts);

- `Company` poate fi sters prin apasarea unui buton cu eticheta `delete` (0.5 pts);

**App.js**

import React, { Component } from 'react'

import CompanyList from './CompanyList'

class App extends Component {

  render() {

    return (

      <div>

        A list of companies

        <CompanyList />

      </div>

    )

  }

}

export default App

**company.js**

import React, { Component } from 'react'

class Company extends Component{

    render(){

        let {item} = this.props

        return(

            <div>{item.name}

            <input type="button" value="delete" onClick={() => this.props.onDelete(item.id)} />

            </div>

        )

    }

}

export default Company

**CompanyList.js**

import React, { Component } from 'react'

import CompanyStore from '../stores/CompanyStore'

import Company from './Company'

class CompanyList extends Component {

    constructor(){

        super()

        this.state = {

            companies : []

        }

        this.deleteCompany = (id) => {

            this.store.deleteOne(id)

        }

    }

    componentDidMount(){

        this.store = new CompanyStore()

        this.setState({

            companies : this.store.getAll()

        })

        this.store.emitter.addListener('UPDATE', () => {

            this.setState({

                companies : this.store.getAll()

            })

        })

    }

  render() {

    return (

      <div>

            {

            this.state.companies.map((e, i) =>

              <Company item={e} key={i} onDelete={this.deleteCompany}/>

            )

        }

      </div>

    )

  }

}

export default CompanyList

**4)React Ana V1 5/5 sau Andrei V1**

**# Subiect 4**

**# Tematica: REACT**

**# Avand urmatoarea aplicatie create folosind `create-react-app`, modificati `Company` astfel incat:**

- aplicatia se deseneaza corect (0.5 pts);

- `CompanyList` se deseneaza ca o lista de  `Company`, iar fiecare `Company` are un buton cu eticheta `edit` (0.5 pts);

- Daca se da click pe butonul edit al unui `Company` aceasta trece in mod de editare (0.5 pts);

- Daca in mod edit se da click pe butonul cancel al unui `Company` aceasta trece in mod view (0.5 pts);

- Se poate salva o companie, iar schimbarea se reflecta in lista de companii (0.5 pts);

**App.js**

import React, { Component } from 'react'

import CompanyList from './CompanyList'

class App extends Component {

  render() {

    return (

      <div>

        A list of companies

        <CompanyList />

      </div>

    )

  }

}

export default App

**Company.js**

import React, { Component } from 'react'

class Company extends Component {

  constructor(props){

    super(props)

    let {item} = this.props

    this.state = {

      name : item.name,

      employees : item.employees,

      revenue : item.revenue,

      isEditing: false

    }

    this.handleChange = (evt) => {

      this.setState({

        [evt.target.name] : evt.target.value

      })

    }

  }

  render() {

    let {item} = this.props

    if (this.state.isEditing){

      return (

        <div>

          <input type="text" id="name" name="name" onChange={this.handleChange} value={this.state.name} />

           <input type="text" id="employees" name="employees" onChange={this.handleChange} value={this.state.employees} />

            <input type="text" id="revenue" name="revenue" onChange={this.handleChange} value={this.state.revenue} />

            <input type="button" value="save" onClick={() => {

              this.props.onSave(item.id, {

                name : this.state.name,

                employees : this.state.employees,

                revenue : this.state.revenue

              })

              this.setState({isEditing : false})

              }

            } />

          <input type="button" value="cancel" onClick={() => this.setState({isEditing : false})} />

        </div>

      )

    }

    else{

      return (

        <div>

          Name {item.name} with {item.employees} employees {item.revenue} revenue

          <input type="button" value="edit" onClick={() => this.setState({isEditing : true})} />

        </div>

      )

    }

  }

}

export default Company

**CompanyList.js**

import React, { Component } from 'react'

import CompanyStore from '../stores/CompanyStore'

import Company from './Company'

class CompanyList extends Component {

    constructor(){

        super()

        this.state = {

            companies : []

        }

        this.saveCompany = (id, company) => {

            this.store.saveOne(id, company)

        }

    }

    componentDidMount(){

        this.store = new CompanyStore()

        this.setState({

            companies : this.store.getAll()

        })

        this.store.emitter.addListener('UPDATE', () => {

            this.setState({

                companies : this.store.getAll()

            })

        })

    }

  render() {

    return (

      <div>

            {

            this.state.companies.map((e, i) =>

                <Company item={e} key={i} onSave={this.saveCompany} />

            )

        }

      </div>

    )

  }

}

export default CompanyList

**5) React Ana V4 5/5**

**# Subiect 4**

**# Tematica: REACT**

**# Avand urmatoarea aplicatie pentru vanzare automata de produse  realizati modificari astfel incat:**

- aplicatia se deseneaza corect (0.5 pts)

- incarcati lista de produse din ProductStore la desenarea componentei `VendingMachine` (0.5 pts)

- adaugati componenta `Product` afisati numele, pretul si un buton cu eticheta buy care apeleaza metoda onBuy (0.5 pts)

- implementati metoda addTokens care incrementeaza numarul de token cu 1 la fiecare apasare de buton (0.5 pts)

- implementati metoda buyProduct care scade numarul de tokeni cu pretul produsului cumparat; daca nu sunt suficienti tokeni nu se realizeaza vanzare (0.5 pts)

**App.js**

import React, { Component } from 'react'

import VendingMachine from './VendingMachine'

class App extends Component {

  render() {

    return (

      <div>

        Vending Machine

        <VendingMachine />

      </div>

    )

  }

}

export default App

**Product.js**

import React, { Component } from 'react'

class Product extends Component {

    render(){

        return(

            <div>

                <h1>{this.props.name}</h1>

                <h1>{this.props.price}</h1>

                <input type="button" value="buy" onClick={() => this.props.onBuy(this.props.price) } ></input>

            </div>

        )

    }

}

export default Product

**VendingMachine.js**

import React, { Component } from 'react'

import Product from './Product'

import ProductStore from '../stores/ProductStore'

class VendingMachine extends Component {

    constructor() {

        super()

        this.state = {

            products: [],

            tokens: 0

        }

        this.addToken = () => {

            this.setState({tokens: this.state.tokens + 1})

        }

        this.buyProduct = (price) => {

            if(this.state.tokens >= price  ){

                this.setState({tokens:this.state.tokens -price})

             }else{

                alert("Nu sunt suficienti tokens")

             }

        }

    }

    componentDidMount(){

        let productStore = new ProductStore()

        this.setState({products: productStore.getAll()})

}

    render() {

        return (

            <div>

                {this.state.products.map((el, index) => <Product key={index} name={el.name} price={el.price} onBuy={this.buyProduct}  />)}

                <div>Tokens: {this.state.tokens}</div>

                <input type="button" value="add token"  onClick={this.addToken}/>

            </div>

        )

    }

}

export default VendingMachine

**Stefi React V0 5/5**

import React, { Component } from 'react'

import RobotStore from '../stores/RobotStore'

import Robot from './Robot'

// TODO : adăugați o componentă Robot

// afișați o componentă Robot pentru fiecare robot din stare

// o componentă robot ar trebui să afișeze un robot și să permită ștergerea lui

// TODO : add a Robot component

// show a Robot component for each robot in the state

// a robot component should show a robot and allow deletion of a robot

class RobotList extends Component {

    constructor(){

        super()

        this.state = {

            robots : []

        }

        this.deleteRobot = (id) => {

            this.store.deleteRobot(id)

        }

    }

    componentDidMount(){

        this.store = new RobotStore()

        this.setState({

            robots : this.store.getRobots()

        })

        this.store.emitter.addListener('UPDATE', () => {

            this.setState({

                robots : this.store.getRobots()

            })

        })

    }

  render() {

    return (

      <div>

        {

             this.state.robots.map((e, i) => <Robot key={i} item={e} onDelete={this.deleteRobot}/>)

        }

      </div>

    )

  }

}

export default RobotList

import React, { Component } from 'react'

class Robot extends Component {

  render() {

    let {item} = this.props

    return (

      <div>

      Hello, my name is {item.name}. I am a {item.type} and weigh {item.mass}

      <input type="button" value="delete" onClick={() => this.props.onDelete(item.id)} />

      </div>

    )

  }

}

export default Robot

**Stefi React V1 5/5**

import React, { Component } from 'react'

import RobotStore from '../stores/RobotStore'

import Robot from './Robot'

import RobotForm from './RobotForm'

// TODO : adăugați o componentă RobotForm

// RobotForm ar să permită adăugarea unui robot

// TODO : add a RobotForm component

// RobotForm should be able to add a robot

class RobotList extends Component {

    constructor(){

        super()

        this.state = {

            robots : []

        }

        this.store = new RobotStore();

        this.add = (robot) => {

            this.store.addRobot(robot)

        }

    }

    componentDidMount(){

        this.store = new RobotStore()

        this.setState({

            robots : this.store.getRobots()

        })

        this.store.emitter.addListener('UPDATE', () => {

            this.setState({

                robots : this.store.getRobots()

            })

        })

    }

    render() {

        return (

            <div>

                {

                    this.state.robots.map((e, i) =>

                        <Robot item={e} key={i} />

                    )

                }

                  <RobotForm onAdd={this.add} />

            </div>

        )

    }

}

export default RobotList

import React from 'react';

import {Component} from 'react';

class RobotForm extends Component {

    constructor(props){

        super(props)

        this.state = {

            id : '',

            type : '',

            name : '',

            mass : ''

        }

        this.handleChange = (evt) => {

            this.setState({

                [evt.target.name] : evt.target.value

            })

        }

    }

     render(){

        return <div>

            <input id="id" name="id" onChange={this.handleChange} />

            <input id="type" name="type" onChange={this.handleChange} />

             <input id="name" name="name" onChange={this.handleChange} />

              <input id="mass" name="mass" onChange={this.handleChange} />

             <input type="button"  value="add" onClick={() => this.props.onAdd({

              id : this.state.id,

              type : this.state.type,

              name : this.state.name,

               mass : this.state.mass,

            })} />

        </div>

    }

}

export default RobotForm;

**Stefi React V2 5/5**

import React, { Component } from 'react'

// TODO : adăugați posibilitatea de a edita un robot

// editarea se face prin intermediul unui robot cu 2 stări, una de vizualizare și una de editare

// TODO : add the posibility to edit a robot

// editing is done via 2 states a view state and an edit state

class Robot extends Component {

    constructor(props){

        super(props);

         this.state = {

            isEditing : false,

            name : this.props.item.name,

            type : this.props.item.type,

            mass : this.props.item.mass

        }

          this.handleChange = (evt) => {

            this.setState({

                [evt.target.name] : evt.target.value

            })

        }

    }

    render() {

        let {item} = this.props

                if (this.state.isEditing){

        return (

            <div>

                   <input type="text"  id="name" name="name" onChange={this.handleChange} value={this.state.name} />

                   <input type="text"  id="type" name="type" onChange={this.handleChange}  value={this.state.type}/>

                   <input type="text"  id="mass" name="mass" onChange={this.handleChange}  value={this.state.mass}/>

                 <input type="button" value="save" onClick={() => {

                            this.props.onSave(item.id, {

                                name : this.state.name,

                                type : this.state.type,

                                mass : this.state.mass

                            })

                            this.setState({isEditing : false})

                            }

                        } />

                <input type="button" value="cancel" onClick={() => this.setState({

                    isEditing : false

                })} />

            </div>

        )

    }

    else{

          return (<div>

                                Hello, my name is {item.name}. I am a {item.type} and weigh {item.mass}

                    <input type="button" value="edit" onClick={() => this.setState({isEditing : true})} />

                </div>)

    }

}

}

export default Robot;

import React, { Component } from 'react'

import RobotStore from '../stores/RobotStore'

import Robot from './Robot'

class RobotList extends Component {

    constructor(){

        super()

        this.state = {

            robots : []

        }

        this.saveRobot = (id, robot) => {

            this.store.saveRobot(id, robot)

        }

    }

    componentDidMount(){

        this.store = new RobotStore()

        this.setState({

            robots : this.store.getRobots()

        })

        this.store.emitter.addListener('UPDATE', () => {

            this.setState({

                robots : this.store.getRobots()

            })

        })

    }

  render() {

    // a robot component should show a robot and allow editing a robot

    return (

      <div>

        {

            this.state.robots.map((e, i) =>

                <Robot item={e} key={i} onSave={this.saveRobot} />

            )

        }

      </div>

    )

  }

}

export default RobotList

**Stefi React V3 ---nu stiu cerinta? Dar ruleaza tot**

**Stefi React V5 – 5/5**

**# Subiect 4**

**# Tematica: REACT**

**# Avand urmatoarea apliatie create folosind `create-react-app`, completati urmatoarele taskuri:**

- Componenta `AddCar` trebuie adaugata in interiorul componentei `CarList`;

- Componenta `AddCar` trebuie sa contina 3 elemente de tip input cu `id-ul` si `name-ul`: `make`, `model`, `price`;

- Componenta `AddCar` trebuie sa contina un element input de tip buton `button` cu valoarea `add car`, folosit pentru a apela metoda `addCar`;

- Componenta `AddCar` din interiorul componentei `CarList` trebuie sa contina in `props` metoda `onAdd`;

- La apasarea butonului `add car` un nou element trebuie afisat in componenta `CarList`;

import React from 'react';

export class AddCar extends React.Component {

    constructor(props){

        super(props);

        this.state = {

            make: '',

            model: '',

            price: ''

        }

    }

    addCar = () => {

        let car = {

            make: this.state.make,

            model: this.state.model,

            price: this.state.price

        };

        this.props.onAdd(car);

    }

    handleChangeModel=(event)=>{

        this.setState({

            model:event.target.value

        });

    }

     handleChangePrice=(event)=>{

        this.setState({

            price:event.target.value

        });

    }

     handleChangeMake=(event)=>{

        this.setState({

            make:event.target.value

        });

    }

    render(){

        return (

            <div>

                <input id="make"  name="make" onChange={this.handleChangeMake} />

                 <input id="model"  name="model" onChange={this.handleChangeModel} />

                 <input id="price"  name="price" onChange={this.handleChangePrice}/>

                 <input type="button" value="add car" onClick={this.addCar}/>

            </div>

        )

    }

}

export default AddCar;

import React from 'react';

import AddCar from './AddCar.js'

export class CarList extends React.Component {

    constructor(){

        super();

        this.state = {

            cars: []

        };

      this.addCar = (car) => {

            var cars1=this.state.cars;

            cars1.push(car);

            this.setState=({

                cars:cars1

            })

        }

    }

    render(){

        return (

            <div>

                {

                    <AddCar onAdd={this.addCar}/>

                }

            </div>

        )

    }

}

**Stefi React V6 – 5/6**

**# Subiect 4**

**# Tematica: REACT**

**# Avand urmatoarea apliatie create folosind `create-react-app`, completati urmatoarele taskuri:**

- Componenta `AddProduct` trebuie adaugata in interiorul componentei `ProductList`;

- Componenta `AddProduct` trebuie sa contina 3 elemente de tip input cu `id-ul` si `name-ul`: `name`, `category`, `price`;

- Componenta `AddProduct` trebuie sa contina un element input de tip buton `button` cu valoarea `add product`, folosit pentru a apela metoda `addProduct`;

- Componenta `AddProduct` din interiorul componentei `ProductList` trebuie sa contina in `props` metoda `onAdd`;

- La apasarea butonului `add product` un nou element trebuie afisat in componenta `ProductList`;

import React from 'react';

import AddProduct from './AddProduct';

export class ProductList extends React.Component {

    constructor(){

        super();

        this.state = {

            products: []

        };

         this.add = (product) => {

            this.store.addProduct(product);

        }

    }

    render(){

        return(

            <div>

             <AddProduct onAdd={this.add} />

            </div>

        )

    }

}

import React from 'react';

export class AddProduct extends React.Component {

    constructor(props){

        super(props);

        this.state = {

            name: '',

            category: '',

            price: ''

        };

       this.handleChange = (evt) => {

            this.setState({

                [evt.target.name] : evt.target.value

            })

        }

    }

    addProduct = () => {

        let product = {

            name: this.state.name,

            category: this.state.category,

            price: this.state.price

        };

        this.props.onAdd(product);

    }

     render(){

        return <div>

            <input id="name" name="name" onChange={this.handleChange} />

            <input id="category" name="category" onChange={this.handleChange} />

             <input id="price" name="price" onChange={this.handleChange} />

             <input type="button"  value="add product" onClick={() => this.props.onAdd({

              name : this.state.name,

              category : this.state.category,

              price : this.state.price,

            })} />

        </div>

    }

}

export default AddProduct;

**Stefi React V7 6/6**

**# Subiect 4**

**# Tematica: REACT**

**# Avand urmatoarea apliatie create folosind `create-react-app`, completati urmatoarele taskuri:**

- Componenta `AddStudent` trebuie adaugata in interiorul componentei `StudentList`;

- Componenta `AddStudent` trebuie sa contina 3 elemente de tip input cu `id-ul` si `name-ul`: `name`, `surname`, `age`;

- Componenta `AddStudent` trebuie sa contina un element input de tip buton `button` cu valoarea `add student`, folosit pentru a apela metoda `addStudent`;

- Componenta `AddStudent` din interiorul componentei `StudentList` trebuie sa contina in `props` metoda `onAdd`;

- La apasarea butonului `add student` un nou element trebuie afisat in componenta `StudentList`;

import React from 'react';

export class AddStudent extends React.Component {

    constructor(props){

        super(props);

        this.state = {

            name: '',

            surname: '',

            age: ''

        };

    }

  handleChange = (evt) => {

            this.setState({

                [evt.target.name] : evt.target.value

            })

        }

    addStudent = () => {

        let student = {

            name: this.state.name,

            surname: this.state.surname,

            age: this.state.age

        };

        this.props.onAdd(student);

    }

    render(){

        return (

            <div>

                <input id="name" name="name" onChange={this.handleChange} />

            <input id="surname" name="surname" onChange={this.handleChange} />

              <input id="age" name="age" onChange={this.handleChange} />

             <input type="button"  value="add student" onClick={this.addStudent} />

            </div>

        )

    }

}

export default AddStudent;

import React from 'react';

import AddStudent from './AddStudent';

export class StudentList extends React.Component {

    constructor(){

        super();

        this.state = {

            students: []

        };

      this.addStudent = (student) => {

            var students1=this.state.students;

            students1.push(student);

            this.setState=({

                students:students1

            })

        }

    }

    render(){

        return (

            <div>

            <AddStudent onAdd={this.addStudent}/>

            </div>

        )

    }

}

**Stefi React V8 1/6**

**# Subject 4**

**# Topic: REACT**

**# Having the following application created with `create-react-app` complete the following tasks:**

- `AddEmployee` component should be rendered inside `EmployeeList` component;

- `AddEmployee` component should contain 3 inputs with `id` and `name`: `name`, `surname`, `experience`;

- `AddEmployee` component should contain an input of type `button` with the value `add employee`, used to trigger `addEmployee` method;

- `AddEmployee` component inside `EmployeeList` should contain a `props` called `onAdd`;

- When pressing `add employee` a new item should be displayed in `EmployeeList` component;

import React from 'react';

export class AddEmployee extends React.Component {

    constructor(props){

        super(props);

        this.state = {

            name: '',

            surname: '',

            experience: ''

        };

    }

    addEmployee = () => {

        let employee = {

            name: this.state.name,

            surname: this.state.surname,

            experience: this.state.experience

        };

        this.props.onAdd(employee);

    }

    render(){

        return (

            <div>

            </div>

        )

    }

}

import React from 'react';

export class EmployeeList extends React.Component {

    constructor(){

        super();

        this.state = {

            employees: []

        };

    }

    render(){

        return(

            <div>

            </div>

        )

    }

}

**Stefi React V9 5/6**

**# Subiect 4**

**# Tematica: REACT**

**# Avand urmatoarea apliatie create folosind `create-react-app`, completati urmatoarele taskuri:**

- Componenta `AddCoupon` trebuie adaugata in interiorul componentei `CouponList`;

- Componenta `AddCoupon` trebuie sa contina 3 elemente de tip input cu `id-ul` si `name-ul`: `category`, `discount`, `availability`;

- Componenta `AddCoupon` trebuie sa contina un element input de tip buton `button` cu valoarea `add coupon`, folosit pentru a apela metoda `addCoupon`;

- Componenta `AddCoupon` din interiorul componentei `CouponList` trebuie sa contina in `props` metoda `onAdd`;

- La apasarea butonului `add coupon` un nou element trebuie afisat in componenta `CouponList`;

import React from 'react';

export class AddCoupon extends React.Component {

    constructor(props){

        super(props);

        this.state = {

            category: '',

            discount: '',

            availability: ''

        };

    }

  handleChange = (evt) => {

            this.setState({

                [evt.target.name] : evt.target.value

            })

        }

    addCoupon = () => {

        let coupon = {

            category: this.state.category,

            discount: this.state.discount,

            availability: this.state.availability

        };

        this.props.onAdd(coupon);

    }

    render(){

        return(

            <div>

               <input id="category" name="category" onChange={this.handleChange} />

            <input id="discount" name="discount" onChange={this.handleChange} />

             <input id="availability" name="availability" onChange={this.handleChange} />

             <input type="button"  value="add coupon" onClick={this.addCoupon} />

            </div>

        )

    }

}

export default AddCoupon;

import React from 'react';

import AddCoupon from './AddCoupon';

export class CouponList extends React.Component {

    constructor(){

        super();

        this.state = {

            coupons: []

        };

           this.addCoupon = (coupon) => {

            var coupons1=this.state.coupons;

            coupons1.push(coupon);

            this.setState=({

                coupons:coupons1

            })

        }

    }

    render(){

        return(

            <div>

                <AddCoupon onAdd={this.addCoupon}/>

            </div>

        )

    }

}

**Stefi React V10 5/5**

**# Subject 4**

**# Topic: REACT**

**# Having the following application created with `create-react-app` complete the following tasks:**

- `AddTask` component should be rendered inside `TaskList` component;

- `AddTask` component should contain 3 inputs with the following properties `id` and `name` having the following values: `task-name`, `task-priority`, `task-duration`;

- `AddTask` component should contain an input of type `button` with the `value` property `add task`, used to trigger `addTask` method;

- `AddTask` component inside `TaskList` should contain a `props` called `taskAdded`;

- When pressing `add task` button a new item should be displayed in `TaskList` component;

import React from 'react';

export default  class AddTask extends React.Component{

    constructor(props){

        super(props);

        this.state = {

            taskName: '',

            taskPriority: 'low',

            taskDuration: 0

        };

    }

  handleChange = (evt) => {

            this.setState({

                [evt.target.name] : evt.target.value

            })

        }

    render(){

        return (

        <div>

                <input id="task-name"  name="task-name" onChange={this.handleChange} />

                 <input id="task-priority"  name="task-priority" onChange={this.handleChange} />

                 <input id="task-duration"  name="task-duration" onChange={this.handleChange}/>

                 <input type="button" value="add task" onClick={this.addTask}/>

        </div>

        );

    }

  addTask = () => {

        let task = {...this.state};

        this.props.taskAdded(task);

    }

}

import React from 'react';

import AddTask from './AddTask';

export default class TaskList extends React.Component {

    constructor(props){

        super(props);

        this.state = {

            data: []

          };

          this.addTask = (task) => {

            var task1=this.state.data;

            task1.push(task);

            this.setState=({

                task:task1

            })

        }

    }

    render() {

        return (

            <div>

             <AddTask taskAdded={this.addTask} />

            </div>

        );

    }

}

**Stefi React V11 5/5**

**# Subject 4**

**# Topic: REACT**

**# Having the following application created with `create-react-app` complete the following tasks:**

- `AddBook` component should be rendered inside `BookList` component;

- `AddBook` component should contain 3 inputs with the following properties `id` and `name` having the following values: `book-title`, `book-type`, `book-price`;

- `AddBook` component should contain an input of type `button` with the `value` property `add book`, used to trigger `handleAdd` method;

- `AddBook` component inside `BookList` should contain a `props` called `itemAdded `;

- When pressing `add book` button a new item should be displayed in `BookList` component;

import React from 'react';

export default class AddBook extends React.Component{

    constructor(props){

        super(props);

        this.state = {

            bookTitle: '',

            bookType: '',

            bookPrice: 0

        };

    }

handleChange = (evt) => {

            this.setState({

                [evt.target.name] : evt.target.value

            })

        }

    render(){

        return (

        <div>

                <input id="book-title"  name="book-title" onChange={this.handleChange} />

                 <input id="book-type"  name="book-type" onChange={this.handleChange} />

                 <input id="book-price"  name="book-price" onChange={this.handleChange}/>

                 <input type="button" value="add book" onClick={this.handleAdd}/>

        </div>

        );

    }

    handleAdd = () => {

        let item = {...this.state};

        this.props.itemAdded(item);

    }

}

import React from 'react';

import AddBook from './AddBook'

export default class BookList extends React.Component {

    constructor(props){

        super(props);

        this.state = {

            data: []

          };

           this.addBook = (book) => {

            var book1=this.state.data;

            book1.push(book);

            this.setState=({

                book:book1

            })

        }

    }

    render() {

        return (

            <div>

             <AddBook itemAdded={this.addBook} />

            </div>

        );

    }

}

**Stefi React V12**

**# Subject 4**

**# Topic: REACT**

**# Having the following application created with `create-react-app` complete the following tasks:**

- `AddTransaction` component should be rendered inside `TransactionList` component;

- `AddTransaction` component should contain 3 inputs with the following properties `id` and `name` having the following values: `transaction-number`, `transaction-type`, `transaction-amount`;

- `AddTransaction` component should contain an input of type `button` with the `value` property `add transaction`,used to trigger `addTransaction` method;

- `AddTransaction` component inside `TransactionList` should contain a `props` called `itemAdded`;

- When pressing `add transaction` button a new item should be displayed in `TransactionList` component;

# **REST**

**Ana Rest v2 0/5 sau V2 Alice sau Andrei V2**

#Subiect 3 (2.5 pts) **0/5**

#TOPIC: REST

**# Dată fiind aplicația `app` completați metoda `GET` la adresa `/ships`:**

- se pot utiliza parametrii de query `page` si `pageSize`

- Daca nu s-a specificat nici pagina si marimea paginii, se vor returna toate navele (0.5 pts)

- Daca s-a specificat pagina, dar nu page size se va presupune ca marimea paginii este 5 si se va returna a n-a pagina de 5 nave (0.5 pts)

- Daca s-au specificat atat pagina cat si marimea unei pagini, se va returna pagina specificata de marimea specificata (0.5 pts)

- Daca s-a specificat o pagina malformata sau o marime de pagina malformata se vor returna toate navele; (0.5 pts)

- Daca pagina specificata este dincolo de ultima nava, se va returna un array vid. (0.5 pts)

**Ana Rest V3**

#Subiect 3 (2.5 pts) **0/5**

#TOPIC: REST

**# Dată fiind aplicația `app` completați metodele `POST` si `DELETE` pe pathurile `/device` si `/device/:id`:**

- `POST /device` returneaza status code 400 si raspuns `{message: "bad request"}` daca `body-ul` este gol. (0.5 pts)

- `POST /device` returneaza status code 400 si raspuns `{message: "bad request"}` daca `pretul < 0`. (0.5 pts)

- `POST /device` returneaza status code 400 si raspuns `{message: "bad request"}` daca `numele are mai putin de 4 caractere`. (0.5 pts)

- `POST /device` returneaza status code 201 si raspuns `{message: "device created"}` daca `body-ul este valid`. (0.5 pts)

- `DELETE /device/:id` returneaza status code 202 si raspuns `{message: "device deleted"}` daca `id-ul` trimis ca request parameter este gasit in baza de date. (0.5 pts)

**Examen Robert 0/5 sau Ana v1 sau Andrei V1**

#Subiect 3 (2.5 pts)

#TOPIC: REST

**# Dată fiind aplicația `app` completați metodele `PUT` si `DELETE` la adresa `/ships/id`:**

- Daca se incerca modificarea unei nave inexistente raspunsul trebuie sa fie `{"message": "not found"}`. Codul de raspuns va fi: `404`; (0.5 pts)

- Daca se incearca modificarea unei nave existente raspunsul trebuie sa fie `{"message": "accepted"}`. Codul de raspuns va fi: `202`; (0.5 pts)

- O cerere get ulterioara la adresa navei editate trebuie sa reflecte modificarile. Codul de raspuns va fi: `200`; (0.5 pts)

- Daca se incearca stergerea unei nave existente raspunsul trebuie sa fie `{"message": "accepted"}`. Codul de raspuns va fi: `202`; (0.5 pts)

- O cerere get ulterioara la adresa navei sterse trebuie sa returneze `{"message": "not found"}`. Codul de raspuns va fi: `404`; (0.5 pts)

**Ana Rest V0 – 3/5**

#Subiect 3 (2.5 pts)

#TOPIC: REST

**# Dată fiind aplicația `app` completați metoda `POST` la adresa `/ships`:**

- Dacă s-a trimis un request cu un corp gol sau nedefinit, se va returna un json cu următorul format: `{"message": "body is missing"}`. Codul de răspuns va fi: `400`; (0.5 pts)

- Dacă din corpul request-ului lipsesc proprietăți se va returna un json cu următorul format: `{"message": "malformed request"}`. Codul de răspuns va fi: `400`; (0.5 pts)

- Deplasamentul trebuie sa fie un numar mai mare ca 1000; în caz contrar se va returna un json cu următorul format: `{"message": "displacement should be over 1000"}`. Codul de răspuns va fi: `400`; (0.5 pts)

- Dacă nava trimisa prin corpul request-ului este valida, va fi adăugata și se va returna un răspuns cu  codul `201`. Corpul răspunsului va fi `{"message": "created"}`;(0.5 pts)

- Dacă se face un request `GET /ships` corpul răspunsului trebuie să conțină 11 `ships`, inclusiv cea adăugata anterior; (0.5 pts)

const express = require('express')

const bodyParser = require('body-parser')

const Sequelize = require('sequelize')

const mysql = require('mysql2/promise')

const DB\_USERNAME = 'root'

const DB\_PASSWORD = 'robert123'

let conn

mysql.createConnection({

    user : DB\_USERNAME,

    password : DB\_PASSWORD

})

.then((connection) => {

    conn = connection

    return connection.query('CREATE DATABASE IF NOT EXISTS tw\_exam')

})

.then(() => {

    return conn.end()

})

.catch((err) => {

    console.warn(err.stack)

})

const sequelize = new Sequelize('tw\_exam', DB\_USERNAME, DB\_PASSWORD,{

    dialect : 'mysql',

    logging: false

})

let Ship = sequelize.define('student', {

    name : Sequelize.STRING,

    portOfSail : Sequelize.STRING,

    displacement : Sequelize.INTEGER

},{

    timestamps : false

})

const app = express()

app.use(bodyParser.json())

app.get('/create', async (req, res) => {

    try{

        await sequelize.sync({force : true})

        for (let i = 0; i < 10; i++){

            let ship = new Ship({

                name : `name${i}`,

                portOfSail : `port ${i}`,

                displacement : 3000 + 10 \* i

            })

            await ship.save()

        }

        res.status(201).json({message : 'created'})

    }

    catch(err){

        console.warn(err.stack)

        res.status(500).json({message : 'server error'})

    }

})

app.get('/ships', async (req, res) => {

    try{

        let ships = await Ship.findAll()

        res.status(200).json(ships)

    }

    catch(err){

        console.warn(err.stack)

        res.status(500).json({message : 'server error'})

    }

})

app.post('/ships', async (req, res) => {

    try{

        //1.

        if(req.body.constructor === Object && Object.keys(req.body).length === 0){

            res.status(400).send({"message": "body is missing"})

        }

        //2.

        if( !(req.body.hasOwnProperty('name'))|| !(req.body.hasOwnProperty('portOfSail')) || !(req.body.hasOwnProperty('displacement')) ) {

          res.status(400).send({"message":"malformed request"})

        }

        //3.

        const deplasament = Object.getOwnPropertyDescriptor(req.body, "displacement")

        if(deplasament.value < 1000){

            res.status(400).send({"message": "displacement should be over 1000"})

        }

        else{

        //4.

        await Ship.create(req.body)

        res.status(201).send({'message':'created'})

        }

    }

    catch(err){

    }

})

module.exports = app

**Ana Rest V4 0/5**

#Subiect 3 (2.5 pts)

#TOPIC: REST

**# Dată fiind aplicația `app` completați metoda `GET` la adresa `/homeworks` si metoda `GET` la adresa `/homeworks/id`:**

- Daca metoda GET /homeworks a fost apelata fara parametri va returna toate temele din baza de date

- Daca metoda GET /homeworks a fost apelata cu parametrul pass = true va returna toate temele care au obtinut o nota mai mare sau egala ca 5

- Daca metoda GET /homeworks/id a fost apelata cu id inexistent returneaza status code `404`

- Daca metoda GET /homeworks/id a fost apelata cu un id valid returneaza un obiect json

- Daca clientul HTTP accepta doar text metoda GET /homeworks/id returneaza doar continutul temei

**Stefi Rest Var 5**

#Subiect 3

#Tematica: REST

**# Avand urmatoa aplicatie dezvoltata in NodeJS, sa se completeze metoda de tip `POST` de pe calea `/products` :**

- Daca nu exista body pentru cererea http, trebuie sa returnati un JSON cu urmatorul format: `{message: "Body is missing"}`. Codul de raspuns trebuie sa fie: `500`;

- Daca body-ul nu respecta formatul unui produs, trebuie sa returnati un JSON cu urmatorul format: `{message: "Invlid body format"}`. Codul de raspuns trebuie sa fie: `500`;

- Pretul unui produs trebuie sa fie mai mare ca 0.In caz contrar trebuie sa returnati un JSON cu urmatorul format: `{message: "Price should be a positive number"}`. Codul de raspuns trebuie sa fie: `500`;

- Daca produsul exista deja in vector, trebuie sa returnati un JSON cu urmatorul format: `{message: "Product already exists"}`.Codul de raspuns trebuie: `500`. Unicitatea se face in functie de nume;

- Daca body-ul are formatul corespunzator, produsul trebuie adaugat in vector si sa returnati un JSON cu urmatorul format: `{message: "Created"}`. Codul de raspuns trebuie sa fie: `201`;

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

app.use(bodyParser.json());

app.use(cors());

app.locals.products = [

    {

        name: "Iphone XS",

        category: "Smartphone",

        price: 5000

    },

    {

        name: "Samsung Galaxy S10",

        category: "Smartphone",

        price: 3000

    },

    {

        name: "Huawei Mate 20 Pro",

        category: "Smartphone",

        price: 3500

    }

];

app.get('/products', (req, res) => {

    res.status(200).json(app.locals.products);

});

app.post('/products', (req, res, next) => {

    if(JSON.stringify(req.body)=='{}')

    {

        res.status(500).json({message:"Body is missing"});

    }

    else{

        if(req.body.name && req.body.category && req.body.price)

        { if(req.body.price >0)

        { var ok=true;

            app.locals.products.forEach((e)=>

            {

                if(e.name== req.body.name)

                { ok=false;

                res.status(500).json({message: "Product already exists"});

                }

            });

           if(ok==true) app.locals.products.push({name: req.body.name, category: req.body.category, price: req.body.price});

            res.status(201).json({message: "Created"});

        }

        else

        {

            res.status(500).json({message: "Price should be a positive number"});

        }

        }

        else

        {

            res.status(500).json({message: "Invalid body format"});

        }

    }

})

module.exports = app;

**Stefi Rest Var 8 5/5**

#Subiect 3

#Tematica: REST

**# Avand urmatoa aplicatie dezvoltata in NodeJS, sa se completeze metoda de tip `POST` de pe calea `/students` :**

- Daca nu exista body pentru cererea http, trebuie sa returnati un JSON cu urmatorul format: `{message: "Body is missing"}`. Codul de raspuns trebuie sa fie: `500`;

- Daca body-ul nu respecta formatul unui student, trebuie sa returnati un JSON cu urmatorul format: `{message: "Invlid body format"}`. Codul de raspuns trebuie sa fie: `500`;

- Varsta unui student trebuie sa fie mai mare ca 0.In caz contrar trebuie sa returnati un JSON cu urmatorul format: `{message: "Age should be a positive number"}`. Codul de raspuns trebuie sa fie: `500`;

- Daca studentul exista deja in vector, trebuie sa returnati un JSON cu urmatorul format: `{message: "Student already exists"}`.Codul de raspuns trebuie: `500`. Unicitatea se face in functie de nume;

- Daca body-ul are formatul corespunzator, studentul trebuie adaugat in vector si sa returnati un JSON cu urmatorul format: `{message: "Created"}`. Codul de raspuns trebuie sa fie: `201`;

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

app.use(bodyParser.json());

app.use(cors());

app.locals.students = [

    {

        name: "Gigel",

        surname: "Popel",

        age: 23

    },

    {

        name: "Gigescu",

        surname: "Ionel",

        age: 25

    }

];

app.get('/students', (req, res) => {

    res.status(200).json(app.locals.students);

});

app.post('/students', (req, res, next) => {

   if(JSON.stringify(req.body)=="{}")

    {

        res.status(500).json({message:'Body is missing'});

    }

    else

    {

        if(req.body.name && req.body.surname && req.body.age)

        {

            if(req.body.age<0)

            {

                res.status(500).json({message: 'Age should be a positive number'});

            }

            else

            {

                var ok=true;

                app.locals.students.forEach(s=>{

                    if(s.name==req.body.name){

                        res.status(500).json({message: 'Student already exists'});

                        ok=false;

                    }

                });

                if(ok==true)

                {

                    app.locals.students.push({name:req.body.name,surname:req.body.surname,age:req.body.age});

                    res.status(201).json({message: 'Created'});

                }

            }

        }

        else

        {

            res.status(500).json({message: 'Invalid body format'});

        }

    }

});

module.exports = app;

**Stefi Rest V9 5/5**

#Subiect 3

#Tematica: REST

**# Avand urmatoa aplicatie dezvoltata in NodeJS, sa se completeze metoda de tip `POST` de pe calea `/cars` :**

- Daca nu exista body pentru cererea http, trebuie sa returnati un JSON cu urmatorul format: `{message: "Body is missing"}`. Codul de raspuns trebuie sa fie: `500`;

- Daca body-ul nu respecta formatul unei masini, trebuie sa returnati un JSON cu urmatorul format: `{message: "Invlid body format"}`. Codul de raspuns trebuie sa fie: `500`;

- Pretul unei masini trebuie sa fie mai mare ca 0.In caz contrar trebuie sa returnati un JSON cu urmatorul format: `{message: "Price should be a positive number"}`. Codul de raspuns trebuie sa fie: `500`;

- Daca masina exista deja in vector, trebuie sa returnati un JSON cu urmatorul format: `{message: "Product already exists"}`.Codul de raspuns trebuie: `500`. Unicitatea se face in functie de model;

- Daca body-ul are formatul corespunzator, masina trebuie adaugata in vector si sa returnati un JSON cu urmatorul format: `{message: "Created"}`. Codul de raspuns trebuie sa fie: `201`;

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

app.use(bodyParser.json());

app.use(cors());

app.locals.cars = [

    {

        make: "BMW",

        model: "X6",

        price: 50000

    },

    {

        make: "Lamborghini",

        model: "Huracan",

        price: 200000

    },

];

app.get('/cars', (req, res) => {

    res.status(200).json(app.locals.cars);

});

app.post('/cars', (req, res, next) => {

    if(JSON.stringify(req.body)=="{}")

    {

        res.status(500).json({message:'Body is missing'});

    }

    else

    {

        if(req.body.make && req.body.model && req.body.price)

        {

            if(req.body.price<0)

            {

                res.status(500).json({message: 'Price should be a positive number'});

            }

            else

            {

                var ok=true;

                app.locals.cars.forEach(s=>{

                    if(s.model==req.body.model){

                        res.status(500).json({message: 'Car already exists'});

                        ok=false;

                    }

                });

                if(ok==true)

                {

                    app.locals.cars.push({make:req.body.make,model:req.body.model,price:req.body.price});

                    res.status(201).json({message: 'Created'});

                }

            }

        }

        else

        {

            res.status(500).json({message: 'Invalid body format'});

        }

    }

})

module.exports = app;

**Stefi Rest V10 3/5**

#Subiect 3 (2.5 pts)

#TOPIC: REST

**# Dată fiind aplicația `app` completați metoda `POST` la adresa `/students`:**

- Dacă s-a trimis un request cu un corp gol sau nedefinit, se va returna un json cu următorul format: `{"message": "body is missing"}`. Codul de răspuns va fi: `400`; (0.5 pts)

- Dacă din corpul request-ului lipsesc proprietăți se va returna un json cu următorul format: `{"message": "malformed request"}`. Codul de răspuns va fi: `400`; (0.5 pts)

- Vârsta trebuie să fie un număr pozitiv; în caz contrar se va returna un json cu următorul format: `{"message": "age should be a positive number"}`. Codul de răspuns va fi: `400`; (0.5 pts)

- Dacă studentul trimis prin corpul request-ului este valid, va fi adăugat și se va returna un răspuns cu  codul `201`. Corpul răspunsului va fi `{"message": "created"}`;(0.5 pts)

- Dacă se face un request `GET /students` corpul răspunsului trebuie să conțină 11 `students`, inclusiv cel adăugat anterior; (0.5 pts)

const express = require('express')

const bodyParser = require('body-parser')

const Sequelize = require('sequelize')

const mysql = require('mysql2/promise')

const DB\_USERNAME = 'root'

const DB\_PASSWORD = 'p@ss'

let conn

mysql.createConnection({

    user : DB\_USERNAME,

    password : DB\_PASSWORD

})

.then((connection) => {

    conn = connection

    return connection.query('CREATE DATABASE IF NOT EXISTS tw\_exam')

})

.then(() => {

    return conn.end()

})

.catch((err) => {

    console.warn(err.stack)

})

const sequelize = new Sequelize('tw\_exam', DB\_USERNAME, DB\_PASSWORD,{

    dialect : 'mysql',

    logging: false

})

let Student = sequelize.define('student', {

    name : Sequelize.STRING,

    address : Sequelize.STRING,

    age : Sequelize.INTEGER

},{

    timestamps : false

})

const app = express()

app.use(bodyParser.json())

app.get('/create', async (req, res) => {

    try{

        await sequelize.sync({force : true})

        for (let i = 0; i < 10; i++){

            let student = new Student({

                name : 'name ' + i,

                address : 'some address on ' + i + 'th street',

                age : 30 + i

            })

            await student.save()

        }

        res.status(201).json({message : 'created'})

    }

    catch(err){

        console.warn(err.stack)

        res.status(500).json({message : 'server error'})

    }

})

app.get('/students', async (req, res) => {

    try{

        let students = await Student.findAll()

        res.status(200).json(students)

    }

    catch(err){

        console.warn(err.stack)

        res.status(500).json({message : 'server error'})

    }

})

app.post('/students', async (req, res) => {

     var obj = JSON.stringify(req.body);

    try{

       if(obj=='{}')

       {

            res.status(400).json({"message": "body is missing"});

       }

       else{

           if(req.body.age && req.body.name && req.body.address ) {

              if(req.body.age>0)

        {

                 Student.build(req.body).save().then(student => {

                    res.status(201).send({

                        message: "created"

                    });

                })

        }

        else

        {

            res.status(400).json({message:"age should be a positive number"})

        }

            }

            else{

               res.status(400).json({"message": "malformed request"});

            }

       }

    }

    catch(err){

        console.warn(err.stack)

        res.status(500).json({message : 'server error'})

    }

})

module.exports = app

**Stefi Rest V11 2/5**

#Subiect 3 (2.5 pts)

#TOPIC: REST

**# Dată fiind aplicația `app` completați metoda `PUT` la adresa `/students/:id`:**

- Dacă s-a trimis un request cu un corp gol sau nedifinit, se va returna un json cu următorul format: `{"message": "body is missing"}`. Codul de răspuns va fi: `400`; (0.5 pts)

- Dacă din corpul request-ului lipsesc proprietăți se va returna un json cu următorul format: `{"message": "malformed request"}`. Codul de răspuns va fi: `400`; (0.5 pts)

- Un student inexistent nu poate fi modificat. Dacă se cere modificarea unui student inexistent se va returna un mesaj cu formatul: `{"message": "not found"}`. Codul de răspuns va fi: `404`; (0.5 pts)

- Dacă studentul există și corpul request-ului este valid, va fi modificat și se va returna un răspuns cu  codul `202`. Corpul răspunsului va fi `{"message": "accepted"}`;(0.5 pts)

- Dacă se face un request `GET /students` corpul răspunsului trebuie să conțină 10 `students`, inclusiv cel modificat anterior; (0.5 pts)

const express = require('express')

const bodyParser = require('body-parser')

const Sequelize = require('sequelize')

const mysql = require('mysql2/promise')

const DB\_USERNAME = 'root'

const DB\_PASSWORD = 'p@ss'

let conn

mysql.createConnection({

    user : DB\_USERNAME,

    password : DB\_PASSWORD

})

.then((connection) => {

    conn = connection

    return connection.query('CREATE DATABASE IF NOT EXISTS tw\_exam')

})

.then(() => {

    return conn.end()

})

.catch((err) => {

    console.warn(err.stack)

})

const sequelize = new Sequelize('tw\_exam', DB\_USERNAME, DB\_PASSWORD,{

    dialect : 'mysql',

    logging: false

})

let Student = sequelize.define('student', {

    name : Sequelize.STRING,

    address : Sequelize.STRING,

    age : Sequelize.INTEGER

},{

    timestamps : false

})

const app = express()

app.use(bodyParser.json())

app.get('/create', async (req, res) => {

    try{

        await sequelize.sync({force : true})

        for (let i = 0; i < 10; i++){

            let student = new Student({

                name : 'name ' + i,

                address : 'some address on ' + i + 'th street',

                age : 30 + i

            })

            await student.save()

        }

        res.status(201).json({message : 'created'})

    }

    catch(err){

        console.warn(err.stack)

        res.status(500).json({message : 'server error'})

    }

})

app.get('/students', async (req, res) => {

    try{

        let students = await Student.findAll()

        res.status(200).json(students)

    }

    catch(err){

        console.warn(err.stack)

        res.status(500).json({message : 'server error'})

    }

})

app.put('/students/:id', async (req, res) => {

   var obj = JSON.stringify(req.body);

    try{

       if(obj=='{}')

       {

            res.status(400).json({"message": "body is missing"});

       }

       else{

           if(req.body.age && req.body.name && req.body.address ) {

             let student = await Student.findByPk(req.params.id)

                if (student){

                    await student.update(req.body)

                res.status(202).json({message : "accepted"})

        }

        else{

            res.status(404).json({message :  "not found"})

        }

            }

            else{

               res.status(400).json({"message": "malformed request"});

            }

       }

    }

    catch(err){

        console.warn(err.stack)

        res.status(500).json({message : 'server error'})

    }

})

module.exports = app

# **SERVICII SIMPLE**

**Alice Sau Ana Simple v2 5/5 (este si Stefi v2 dar e 4/5)**

**# Subiect 1 (2.5 pts) 5/5**

**# Tematica: Clienți și servere simple**

**# Dat fiind serverul `server.js` și fișierul `index.html` din directorul `public` directory:**

**# Satisfaceți următoarele cerințe:**

- fișierul `index.html`, care conține textul `A simple app` este livrat de server ca conținut static (0.5 pts);

- butonul cu id-ul `del` există în pagină și se poate da click pe el(0.5 pts);

- la încărcarea paginii sunt încărcate toate elementele în tabelul cu id `main` cu un `tr` pentru fiecare (0.5 pts);

- când se dă click pe butonul `del` se șterg elementele cu numele prezent în text input-ul cu id-ul `name` (0.5 pts);

- elementele cu numele selectat pentru ștergere nu mai apar în tabel (0.5 pts);

const express = require('express')

const app = express()

app.use(express.static('public'))

app.locals.cars = [{

    name :  'a',

    color : 'red'

},{

    name :  'b',

    color : 'blue'

}]

app.get('/cars', (req, res) => {

    res.status(200).json(app.locals.cars)

})

app.delete('/cars/:name', (req, res) => {

    app.locals.cars = app.locals.cars.filter((e) => e.name !== req.params.name)

    res.status(202).json({message : 'accepted'})

})

module.exports = app

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>A simple app</title>

    <script>

        window.onload = async () => {

            let btn = document.getElementById('del')

            let name = document.getElementById('name')

            btn.onclick = async () => {

                await deleteCar(name.value)

            }

            let deleteCar = async (name) => {

                try {

                    await fetch(`/cars/${name}`, {

                        method: 'delete'

                    }).then((response) => {

                        return response.json();

                    }).then(() => load());

                }

                catch (err) {

                    console.warn(err)

                }

            }

            let tbl = document.getElementById('main')

            let load = async () => {

                try {

                    let response = await fetch(`/cars`)

                    let data = await response.json()

                    let rows = data.map((e) => `

                        <tr>

                            <td>

                                ${e.name}

                            </td>

                            <td>

                                ${e.color}

                            </td>

                        </tr>

                    `)

                    console.warn(rows)

                    tbl.innerHTML = rows.join('')

                }

                catch (err) {

                    console.warn(err)

                }

            }

            load();

        }

    </script>

</head>

<body>

    A simple app

    <input type="text" placeholder="name" id="name" />

    <table id=main></table>

    <button id="del">Delete</button>

</body>

</html>

**2) Var 3 Alice sau Ana V3 3/5**

**# Subiect 1 (2.5 pts)**

**# Tematica: Clienți și servere simple**

**# Dat fiind serverul `app.js` și fișierul `index.html` din directorul `public`:**

**# Satisfaceți următoarele cerințe:**

- Fișierul `index.html`, trebuie sa contina un element de tip `paragraf` cu textul `Webtech app` si sa fie furnizat ca si continut static din directorul `public` (0.5 pts);

- Butoanele cu id-urile `load` si `delete` există în pagină și nu se afla intr-o stare disabled (0.5 pts);

- La apasarea butonului `load` sunt încărcate toate elementele din fisierul `data.json` în tabelul cu id `table` cu un `tr` pentru fiecare element si 3 `td-uri` pentru fiecare proprietate (0.5 pts);

- Input-ul de tip `text` cu id-ul `name` exista in pagina (0.5pts);

- La apasarea butonului `delete` se șterg elementele cu numele prezent în text input-ul cu id-ul `name` (0.5 pts);

<!DOCTYPE html>

<html>

<head>

    <title>Webtech</title>

</head>

<body>

    <p>Webtech app</p>

    <table id="table"></table>

    <button id="load">load</button>

    <button id="delete">delete</button>

    <input type="text" id="name"></input>

    <script>

        window.onload = async () => {

            let btnload = document.getElementById('load')

            let btndelete = document.getElementById('delete')

            let tbl = document.getElementById('table')

            let name = document.getElementById('name')

            let people

            let getPeople = async () => {

                try {

                    const response = await fetch('../data.json');

                    const json = await response.json();

                    return json;

                }

                catch (err) {

                    console.log(err);

                }

            }

            btnload.addEventListener("click", async () => {

                try {

                    getPeople().then(data => {

                        people = data

                        let rows = data.map((e) => `

                            <tr>

                                <td>

                                    ${e.name}

                                </td>

                                <td>

                                    ${e.surname}

                                </td>

                                <td>

                                    ${e.age}

                                </td>

                            </tr>

                        `)

                        tbl.innerHTML = rows.join('')

                    })

                }

                catch (err) {

                    console.warn(err)

                }

            })

            btndelete.addEventListener("click", async () => {

                if (people) {

                    people = people.filter(e => e.name !== name.value)

                    let rows = people.map((e) => `

                            <tr>

                                <td>

                                    ${e.name}

                                </td>

                                <td>

                                    ${e.surname}

                                </td>

                                <td>

                                    ${e.age}

                                </td>

                            </tr>

                        `)

                    console.warn(rows)

                    tbl.innerHTML = rows.join('')

                }

            })

        }

    </script>

</body>

</html>

const express = require('express');

const cors = require('cors');

const path = require('path');

const app = express();

app.use(express.static('public'))

app.use(cors());

module.exports = app;

**3) Examen Robert 5/5 sau Ana Simple V1 sau Andrei V1**

**# Subiect 1 (2.5 pts)**

**# Tematica: Clienți și servere simple**

**# Dat fiind serverul `server.js` și fișierul `index.html` din directorul `public` directory:**

**# Satisfaceți următoarele cerințe:**

- fișierul `index.html`, care conține textul `A simple app` este livrat de server ca conținut static (0.5 pts);

- butonul cu id-ul `reload` există în pagină și se poate da click pe el(0.5 pts);

- când se dă click pe butonul cu id-ul `reload` nimic in filtru, toate elementele sunt returnate (0.5 pts);

- când se dă click pe butonul cu id-ul `reload` cu valoarea `red` in filtru, elementele roșii sunt returnate (0.5 pts);

- când se dă click pe butonul cu id-ul `reload` cu o valoare in filtru care nu corespunde culorii niciunui element, este returnată o listă vidă (0.5 pts);

const express = require('express')

const app = express()

app.use(express.static('public'))

app.get('/cars', (req, res) => {

    let filter = req.query.filter

    if (!filter){

        res.status(200).json([{

            name :  'a',

            color : 'red'

        },{

            name :  'b',

            color : 'blue'

        }])

    }

    else{

        res.status(200).json([{

            name :  'a',

            color : 'red'

        },{

            name :  'b',

            color : 'blue'

        }].filter((e) => e.color === filter))

    }

})

module.exports = app

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>A simple app</title>

    <script>

        window.onload = async () => {

            let btn = document.getElementById('reload')

            let filter = document.getElementById('filter')

            let tbl = document.getElementById('main')

            btn.addEventListener("click", () => {

                const filterValue = filter.value;

                load(filterValue);

            })

            let load = async (filter) => {

                try {

                    const response = await fetch(`http://localhost:8080/cars?filter=${filter}`);

                    const data = await response.json();

                    console.log(data)

                    if (data.length > 0) {

                        let rows = data.map((e) => `

                        <tr>

                            <td>

                                ${e.name}

                            </td>

                            <td>

                                ${e.color}

                            </td>

                        </tr>

                    `)

                        tbl.innerHTML = rows.join('')

                    } else

                        tbl.innerHTML = [];

                }

                catch (err) {

                    console.warn(err)

                }

            }

            load('')

        }

    </script>

</head>

<body>

    A simple app

    <input type="text" placeholder="filter" id="filter" />

    <table id=main></table>

    <button id="reload">Reload</button>

</body>

</html>

**Sau**

**Stefi Simple Var 1 5/5**

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>A simple app</title>

</head>

<body>

    A simple app

    <table id=main></table>

    <input type="text" id="filter" />

    <button id=reload></button>

</body>

<script>

    window.onload = async () => {

        let btn = document.getElementById('reload')

        let filter = document.getElementById('filter')

        let tbl = document.getElementById('main')

        const preluareDate =()=>{

            const getData = new XMLHttpRequest();

            getData.open('GET', 'http://localhost:8080/cars')

            getData.responseType = 'json'

            getData.onload =()=>{

                const myData = getData.response

                console.log(filter.value)

                if(filter.value.length == 0){

                    for(let i=0;i<myData.length;i++){

                        let linie = document.createElement('tr')

                        for(let element in myData[i]){

                            let coloana = document.createElement('td')

                            coloana.innerText = myData[i][element]

                            linie.append(coloana)

                        }

                        tbl.append(linie)

                    }

                }

                else if(filter.value == 'red'){

                    for(let i=0;i<myData.length;i++){

                        if(myData[i]['color'] == 'red'){

                            let linie = document.createElement('tr')

                            for(let element in myData[i]){

                                let coloana = document.createElement('td')

                                coloana.innerText = myData[i][element]

                                linie.append(coloana)

                            }

                            tbl.append(linie)

                        }

                    }

                }

            }

            getData.send()

        }

        btn.addEventListener('click', preluareDate)

    }

</script>

</html>

**4) Ana Simple V0 5/5 sau Andrei V0**

**# Subiect 1 (2.5 pts)**

**# Tematica: Clienți și servere simple**

**# Dat fiind serverul `app.js` și fișierul `index.html` din directorul `public` directory:**

**# Satisfaceți următoarele cerințe:**

- fișierul `index.html`, care conține textul `A simple app` este livrat de server ca conținut static (0.5 pts);

- butonul cu id-ul `load` există în pagină și se poate da click pe el(0.5 pts);

- când se dă click pe butonul cu id-ul `load` se cere lista de `cars` de pe server; mașinile cu culoare `red` sunt încărcate în tabelul cu id `main` cu un `tr` pentru fiecare (0.5 pts);

- tabelul conține câte un `tr` pentru fiecare mașină încărcată (0.5 pts);

- sunt afișate doar mașini cu culoarea `red` (0.5 pts);

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>A simple app</title>

    <script>

        function clickBtn(){

            fetch('http://localhost:8080/cars').then(result =>result.json()).then(data =>{

                //console.log(data)

                let table = document.getElementById('main')

                for(let i=0;i<data.length;i++){

                    if(data[i].color === 'red'){

                        let tr = document.createElement('tr')

                        let td = document.createElement('td')

                        td.innerText = data[i].name

                        tr.append(td)

                        table.append(tr)

                    }

                }

            })

        }

        function app(){

            let btnLoad = document.getElementById('load')

            btnLoad.addEventListener('click', clickBtn)

        }

        document.addEventListener("DOMContentLoaded",app)

    </script>

</head>

<body>

    A simple app

    <table id="main"></table>

    <button id="load">LOAD</button>

</body>

</html>

const express = require('express')

const app = express()

app.use("/",express.static("./public"))

app.get('/cars', (req, res) => {

    res.status(200).json([{

        name :  'a',

        color : 'red'

    },{

        name :  'b',

        color : 'blue'

    }])

})

module.exports = app

**5) Ana Simple V4 5/5**

**# Subiect 1 (2.5 pts)**

**# Tematica: Clienți și servere simple**

**# Dat fiind serverul `app.js` și fișierul `index.html` din directorul `public` directory:**

**# Satisfaceți următoarele cerințe:**

- fișierul `index.html`, care conține textul `A simple app` este livrat de server ca conținut static (0.5 pts);

- fișierul `profil.json` livrat de catre server respecta structura descrisa in test (0.5 pts)

- pagina `index.html` conține titlu de rang 1 cu textul `Profil Influencer` (0.5 pts)

- detaliile profiluli (name, instagram, youtube) sunt afisate in paragrafe separate in div-ul cu id=content (0.5 pts)

- butonul convert afiseaza numarul de urmaritori pe instagram in milioane (ex: 5M); operatia se realizeaza doar pe client (0.5 pts)

**Index.html**

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>A simple app</title>

    <script>

            let load = async () => {

                try{

                    let response = await fetch(`/profile.json`)

                    let object = await response.json()

                    let content = ''

                    content+= `

                        <p id="p1"> ${object.name} </p>

                        <p id="p2"> ${object.instagram} </p>

                        <p id="p3"> ${object.youtube} </p>

                    `

                    document.getElementById("content").innerHTML = content;

                }

                catch(err){

                    console.warn(err)

                }

            }

            let convert = () => {

                let instagram = document.getElementById('p2').innerHTML

                document.getElementById("p2").innerHTML = instagram/10000000 + "M";

            }

            document.addEventListener('DOMContentLoaded', load)

    </script>

</head>

<body>

    <h1>Profil influencer</h1>

    A simple app

    <div id="content"></div>

    <input type="button" value="convert" id="convert" onclick="convert()" />

</body>

</html>

**Profile.json**

{

    "name": "influencer",

    "instagram": 1000000,

    "youtube": 2000000

}