

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Звіт з лабораторних робіт**

з дисципліни “Програмне забезпечення систем автоматичної ідентифікації”

Варіант №5

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент V курсу  групи КП-11мн  Кравчук Аркадій Андрійович |  |  | Перевірила  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2021 р.  викладач  Боярінова Юлія Євгеніївна |

Київ 2021

**Мета роботи**

Набуття студентами практичного досвіду з розроблення штрихових кодів з оптимізованими параметрами.

**Завдання**

Варіант №5 за списком. Штриховий код – EAN-13.

*Лабораторна робота №1*. Розробка архітектури бази даних КСАІ. Завдання: розробити архітектуру бази даних КСАІ для збереження інформації про одиниці обліку, розробити СУБД.

*Лабораторна робота №2*. Розробка інтерфейсу користувача КСАІ. Завдання: розробити дизайн та інтерфейс користувача КСАІ.

*Лабораторна робота №3*. Розробка програми кодування вихідної інформації КСАІ. Завдання: розробити алгоритм та програму штрихового кодування інформації про одиниці обліку (використовуючи результат виконання розрахункової роботи).

*Лабораторна робота №4*. Розробка програми “друкування” (зберігання у вигляді файлу) штрих кодової позначки . Завдання: розробити алгоритм та програму «друкування» штрих кодової позначки (з урахуванням особливостей одиниці обліку).

*Лабораторна робота №5*. Розробка програми декодування штрих кодової позначки. Завдання: розробити алгоритм та програму декодування штрих кодової позначки.

*Лабораторна робота №6*. Розробка та налагодження пакету програм КСАІ. Завдання: розробити та налагодити пакет програм КСАІ.

**Код програмного забезпечення**

Лістинг програмного модулю для кодування та декодування штрих-кодової позначки:

|  |
| --- |
| app/ean13.ts |
| // EAN13  class EAN13 {  private static H1 = [1, 0, 1];  private static H4 = [0, 1, 0, 1, 0];  private static A = [  [0, 0, 0, 1, 1, 0, 1],  [0, 0, 1, 1, 0, 0, 1],  [0, 0, 1, 0, 0, 1, 1],  [0, 1, 1, 1, 1, 0, 1],  [0, 1, 0, 0, 0, 1, 1],  [0, 1, 1, 0, 0, 0, 1],  [0, 1, 0, 1, 1, 1, 1],  [0, 1, 1, 1, 0, 1, 1],  [0, 1, 1, 0, 1, 1, 1],  [0, 0, 0, 1, 0, 1, 1]  ];  private static B = [  [0, 1, 0, 0, 1, 1, 1],  [0, 1, 1, 0, 0, 1, 1],  [0, 0, 1, 1, 0, 1, 1],  [0, 1, 0, 0, 0, 0, 1],  [0, 0, 1, 1, 1, 0, 1],  [0, 1, 1, 1, 0, 0, 1],  [0, 0, 0, 0, 1, 0, 1],  [0, 0, 1, 0, 0, 0, 1],  [0, 0, 0, 1, 0, 0, 1],  [0, 0, 1, 0, 1, 1, 1]  ];  private static C = [  [1, 1, 1, 0, 0, 1, 0],  [1, 1, 0, 0, 1, 1, 0],  [1, 1, 0, 1, 1, 0, 0],  [1, 0, 0, 0, 0, 1, 0],  [1, 0, 1, 1, 1, 0, 0],  [1, 0, 0, 1, 1, 1, 0],  [1, 0, 1, 0, 0, 0, 0],  [1, 0, 0, 0, 1, 0, 0],  [1, 0, 0, 1, 0, 0, 0],  [1, 1, 1, 0, 1, 0, 0]  ];  private static first\_half = [  [EAN13.A, EAN13.A, EAN13.A, EAN13.A, EAN13.A, EAN13.A],  [EAN13.A, EAN13.A, EAN13.B, EAN13.A, EAN13.B, EAN13.B],  [EAN13.A, EAN13.A, EAN13.B, EAN13.B, EAN13.A, EAN13.B],  [EAN13.A, EAN13.A, EAN13.B, EAN13.B, EAN13.B, EAN13.A],  [EAN13.A, EAN13.B, EAN13.A, EAN13.A, EAN13.B, EAN13.B],  [EAN13.A, EAN13.B, EAN13.B, EAN13.A, EAN13.A, EAN13.B],  [EAN13.A, EAN13.B, EAN13.B, EAN13.B, EAN13.A, EAN13.A],  [EAN13.A, EAN13.B, EAN13.A, EAN13.B, EAN13.A, EAN13.B],  [EAN13.A, EAN13.B, EAN13.A, EAN13.B, EAN13.B, EAN13.A],  [EAN13.A, EAN13.B, EAN13.B, EAN13.A, EAN13.B, EAN13.A],  ];  private static first\_half\_is\_A = [  [true, true, true, true, true, true ],  [true, true, false, true, false, false],  [true, true, false, false, true, false],  [true, true, false, false, false, true ],  [true, false, true, true, false, false],  [true, false, false, true, true, false],  [true, false, false, false, true, true ],  [true, false, true, false, true, false],  [true, false, true, false, false, true ],  [true, false, false, true, false, true ],  ];  private source\_number: string;  private source\_number\_arr: number[];  private check\_digit: number;  private barcode\_arr: number[];  public constructor(source\_number: string) {  this.source\_number = source\_number;  this.source\_number\_arr = [];  for (let i = 0; i < source\_number.length; i++) {  this.source\_number\_arr.push(parseInt(source\_number[i]));  }  this.check\_digit = this.calculateCheckDigit();  this.barcode\_arr = this.calculatBarcode();  }  private calculatBarcode() {  const res: number[] = [];  const check\_digit = this.check\_digit;  const cur\_first\_half = EAN13.first\_half[check\_digit];  for (let i = 0; i < EAN13.H1.length; i++) {  res.push(EAN13.H1[i]);  }  for (let i = 12; i > 6; i--) {  const cur\_digit\_arr = cur\_first\_half[12 - i];    const cur\_digit = this.source\_number\_arr[12 - i];  const vals\_for\_digit = cur\_digit\_arr[cur\_digit];  for (let j = 0; j < vals\_for\_digit.length; j++) {  res.push(vals\_for\_digit[j]);  }  }  for (let i = 0; i < EAN13.H4.length; i++) {  res.push(EAN13.H4[i]);  }  for (let i = 6; i < 12; i++) {  const cur\_digit = this.source\_number\_arr[i];  const vals\_for\_digit = EAN13.C[cur\_digit];  for (let j = 0; j < vals\_for\_digit.length; j++) {  res.push(vals\_for\_digit[j]);  }  }  for (let i = 0; i < EAN13.H1.length; i++) {  res.push(EAN13.H1[i]);  }  return res;  }  private calculateCheckDigit() {  let odd\_sum = 0;  let even\_sum = 0;  for (let i = 0; i < this.source\_number\_arr.length; i++) {  if (i % 2 == 0) {  even\_sum += this.source\_number\_arr[i];  } else {  odd\_sum += this.source\_number\_arr[i];  }  }  const sum = (even\_sum \* 33) + odd\_sum;  return (10 - (sum % 10)) % 10;  }  public getSourceNumber() {  return this.source\_number;  }  public getBarcodeArr() {  return this.barcode\_arr;  }  public getCountryCode() {  return this.source\_number.substr(0, 3);  }  public getManufacturerCode() {  return this.source\_number.substr(3, 4);  }  public getProdcutCode() {  return this.source\_number.substr(7, 5);  }  public getCheckDigit() {  return this.check\_digit;  }  public static decode(digits: string) {  if (digits.length != 95) {  return null;  }  let i = 0;  for (; i < EAN13.H1.length; i++) {  if ((+digits[i]) !== EAN13.H1[i]) {  return null;  }  }  const cur\_first\_half\_check\_is\_A = [];  const cur\_first\_half\_postion = [];  for (let j = 0; j < EAN13.first\_half[0].length; j++) {  const digits\_block = digits.substr(i, EAN13.first\_half[0][0][0].length);  i += EAN13.first\_half[0][0][0].length;  let found\_in\_A = false;  for (let k = 0; k < EAN13.A.length && !found\_in\_A; k++) {  const check\_block = EAN13.A[k].join("");  if (digits\_block === check\_block) {  cur\_first\_half\_check\_is\_A.push(true);  cur\_first\_half\_postion.push(k);  found\_in\_A = true;  break;  }  }  let found\_in\_B = false;  for (let k = 0; k < EAN13.B.length && !found\_in\_A && !found\_in\_B; k++) {  const check\_block = EAN13.B[k].join("");  if (digits\_block === check\_block) {  cur\_first\_half\_check\_is\_A.push(false);  cur\_first\_half\_postion.push(k);  found\_in\_B = true;  break;  }  }  if (!found\_in\_A && !found\_in\_B) {  return null;  }  }  let check\_digit = -1;  for (let j = 0; j < this.first\_half\_is\_A.length; j++) {  const check\_arr = this.first\_half\_is\_A[j];  if (check\_arr.join() == cur\_first\_half\_check\_is\_A.join()) {  check\_digit = j;  break;  }  }  if (check\_digit < 0) {  return null;  }  const reversed\_barcode = [ ...cur\_first\_half\_postion ];  for (let j = 0; j < EAN13.H4.length; j++) {  const check\_ = +digits[i++];  if (EAN13.H4[j] !== check\_) {  return null;  }  }  for (let j = 0; j < EAN13.first\_half[0].length; j++) {  const last\_digits\_block = digits.substr(i, EAN13.C[0].length);  i += EAN13.C[0].length;  let found\_in\_C = false;  for (let k = 0; k < EAN13.C.length && !found\_in\_C; k++) {  const check\_block = EAN13.C[k].join("");  if (last\_digits\_block === check\_block) {  reversed\_barcode.push(k);  found\_in\_C = true  break;  }  }  if (found\_in\_C != true) {  return null;  }  }  for (; i < EAN13.H1.length; i++) {  if ((+digits[i]) !== EAN13.H1[i]) {  return null;  }  }  if (reversed\_barcode.length != 12) {  return null;  }  const barcode\_normal = [ ...reversed\_barcode ];  const barcode\_normal\_str = barcode\_normal.join("");  const barcode\_obj = new EAN13(barcode\_normal\_str);  if (barcode\_obj.getCheckDigit() !== check\_digit) {  return null;  }  return barcode\_normal\_str;  }  }  export default EAN13; |

Лістинг програмного модулю формування штрих-кодової позначки із вводу користувача з функцією збереження штрих-коду в файл:

|  |
| --- |
| app/encode.tsx |
| import React from "react";  import DigitsPanel from "./digits";  import EAN13 from "./ean13";  import { saveAs } from 'file-saver';  import { IProduct } from "./../products";  import { IManufacture } from "./../manufacturers";  import { ICountry } from "./../countries";  const BARCODE\_DIGITS\_COUNT = 12;  const BARCODE\_NAME = "EAN-13";  type EncodeState = {  barcode\_source\_number\_val: string,  barcode\_source\_number\_is\_valid: boolean,  barcode\_obj: EAN13 | null,  current\_page: "encode" | "decode" | "products",  is\_product\_input: boolean,  product\_current\_idx: number,  product\_name\_input: string,  products: IProduct[],  manufactures: IManufacture[],  countries: ICountry[]  };  class Encode extends React.Component {  state: EncodeState = {  barcode\_source\_number\_val: "",  barcode\_source\_number\_is\_valid: false,  barcode\_obj: null,  current\_page: "encode",  is\_product\_input: false,  product\_name\_input: "",  product\_current\_idx: -1,  products: [],  manufactures: [],  countries: []  }  componentDidMount() {  window.require('electron').ipcRenderer.on('get\_all\_products\_reply', (\_, arg: EncodeState["products"]) => {  this.setState({  products: arg  });  });  window.require('electron').ipcRenderer.send('get\_all\_products');  window.require('electron').ipcRenderer.on('get\_all\_manufacturers\_reply', (\_, arg: EncodeState["manufactures"]) => {  this.setState({  manufactures: arg  });  });  window.require('electron').ipcRenderer.send('get\_all\_manufacturers');  window.require('electron').ipcRenderer.on('get\_all\_countries\_reply', (\_, arg: ICountry[]) => {  this.setState({  countries: arg  });  });  window.require('electron').ipcRenderer.send('get\_all\_countries');  }  change\_page = (is\_product\_input: boolean) => {  if (is\_product\_input === this.state.is\_product\_input) {  return;  }  this.setState({is\_product\_input});  this.reset\_barcode\_source\_number\_val();  this.setState({  product\_name\_input: "",  product\_current\_idx: -1  })  }  remove\_focus\_from\_last\_digit\_if\_needed() {  const last\_digit = document.getElementById("last-input-digit");  if (document.activeElement === last\_digit) {  last\_digit.blur();  }  }  check\_if\_barcode\_source\_number\_val\_is\_valid(val: string) {  return val.length == BARCODE\_DIGITS\_COUNT && /^[0-9]{12}$/.test(val);  }  handleChange\_barcode\_source\_number\_val = (val: string) => {  const is\_valid = this.check\_if\_barcode\_source\_number\_val\_is\_valid(val);  this.setState({  barcode\_source\_number\_val: val,  barcode\_source\_number\_is\_valid: is\_valid  });  if (is\_valid) {  this.setState({  barcode\_obj: new EAN13(val)  });  this.remove\_focus\_from\_last\_digit\_if\_needed();  }  }  reset\_barcode\_source\_number\_val = () => {  this.setState({  barcode\_source\_number\_val: "",  barcode\_source\_number\_is\_valid: false  });  this.make\_focus\_on\_first\_digit\_if\_needed();  }  make\_focus\_on\_first\_digit\_if\_needed() {  const first\_digit = document.getElementById("first-input-digit");  if (document.activeElement !== first\_digit) {  first\_digit?.focus();  }  }  append\_results\_block\_if\_needed = () => {  if (this.state.barcode\_source\_number\_is\_valid) {  return (  <div>  <hr style={{ margin: "0 30px" }}/>  <p style={{ textAlign: "center", marginTop: "10px", marginBottom: "3px", fontSize: "20px", fontWeight: 500 }}>  Results:  </p>  <div style={{ display: "flex", justifyContent: "center", alignItems: "center", flexDirection: "column", marginBottom: "20px" }}>  <canvas width="219px" height="100px" id="barcode-canva" style={{width: "219px", height: "100px", marginBottom: "10px"}}></canvas>  <button className="btn btn-success" onClick={this.saveToFile}><i className="bi bi-download"></i> Save barcode as image (.png)</button>  </div>  <p style={{ marginBottom: "3px", textAlign: "center"}}>Also binary output:</p>  <textarea readOnly={true} value={this.state.barcode\_obj.getBarcodeArr().join("")} style={{ fontFamily: "monospace", width: "920px", height: "30px", resize: "none", display: "block", marginLeft: "auto", marginRight: "auto" }}></textarea>  <button type="button" style={{display: "block"}} className="btn btn-outline-success btn-sm mx-auto my-2" onClick={this.saveToFileAsText}><i className="bi bi-download"></i> Save barcode as binary digits (.txt)</button>  <p style={{ marginTop: "20px", marginBottom: "3px", textAlign: "center"}}>Structured information from barcode:</p>  <pre className="mx-auto" style={{ width: "280px", color: "black", lineHeight: "14px", overflow: 'hidden' }}>  <b>Country code:</b> {this.state.barcode\_obj.getCountryCode()}<br/>  <b>Manufacturer code:</b> {this.state.barcode\_obj.getManufacturerCode()}<br/>  <b>Prodcut code:</b> {this.state.barcode\_obj.getProdcutCode()}<br/>  <b>Check digit:</b> {this.state.barcode\_obj.getCheckDigit()}  </pre>  </div>  );  } else {  return (  <div style={{ display: "flex", flexDirection: "column", alignItems: "center", justifyContent: "center", alignContent: "center" }}>  <p style={{ textAlign: "center", fontSize: "20px" }}>No results...</p>  <p style={{ textAlign: "center", marginTop: "0px", fontStyle: "italic", color: "GrayText" }}>  Please, enter the source number in the input for the purpose of getting here results – the image of barcode  </p>  </div>  );  }  }  componentDidUpdate() {  if (this.state.barcode\_source\_number\_is\_valid) {  const canvas = document.getElementById('barcode-canva') as HTMLCanvasElement;  if (canvas && canvas.getContext) {  const barcode\_arr = this.state.barcode\_obj.getBarcodeArr();  const ctx = canvas.getContext('2d');  const i\_width = 2;  const height = 70;    ctx.fillStyle = "white";  ctx.fillRect(0, 0, 219, 100);  ctx.fillStyle = "black";  for (let i = 0; i < barcode\_arr.length; i++) {  if (barcode\_arr[i] == 1) {  let tmp\_height = height;  if ((i >= 0 && i <= 2) || (i >= 45 && i <= 49) || (i >= 92 && i <= 94)) {  tmp\_height += 15;  }  ctx.fillRect(14 + i\*i\_width, 5+ 0, i\_width, tmp\_height);  }  }  const check\_digit = this.state.barcode\_obj.getCheckDigit();  const barcode = this.state.barcode\_obj.getSourceNumber() + check\_digit.toString();  ctx.font = "20px Arial";  ctx.fillText(barcode[0], 2+ 0, 5+ 88);  for (let i = 1; i < 7; i++) {  ctx.fillText(barcode[i], 2+ 7 + i \* 14, 5+ 88);  }  for (let i = 7; i < 13; i++) {  ctx.fillText(barcode[i], 2+ 15 + i \* 14, 5+ 88);  }  }  }  }  saveToFile = () => {  const canvas = document.getElementById('barcode-canva') as HTMLCanvasElement;  canvas.toBlob((blob) => {  saveAs(blob, "barcode\_" + this.state.barcode\_obj.getSourceNumber() + ".png");  });  }  saveToFileAsText = () => {  saveAs(new Blob([this.state.barcode\_obj.getBarcodeArr().join("")], {type: "text/plain;charset=utf-8"}), "barcode\_" + this.state.barcode\_obj.getSourceNumber() + ".txt", { autoBom: true });  }  get\_text\_input = () => {  return (  <>  <p style={{ marginTop: "20px", textAlign: "center", fontSize: "20px"}}>  Enter <i>{BARCODE\_DIGITS\_COUNT}</i>-digit number in input below:  </p>  <DigitsPanel count\_digits={BARCODE\_DIGITS\_COUNT} value={this.state.barcode\_source\_number\_val} on\_value\_change={this.handleChange\_barcode\_source\_number\_val} />  <pre className="source-barcode">  Entered barcode: "<span><b>{this.state.barcode\_source\_number\_val}</b></span>"  </pre>  <p className="source-barcode-actions">  <button className="btn btn-secondary" onClick={this.reset\_barcode\_source\_number\_val}>Reset barcode input</button>  </p>  </>  );  }  handle\_set\_product = (val: string) => {  const idx = this.state.products.findIndex(x => x.name === val);  this.setState({  product\_name\_input: val,  product\_current\_idx: idx  });  if (idx > 0) {  window.require('electron').ipcRenderer.on('get\_product\_code\_reply', (\_, arg: string) => {  console.log(arg);  this.setState({  barcode\_source\_number\_val: arg,  barcode\_source\_number\_is\_valid: true,  barcode\_obj: new EAN13(arg)  });  });  window.require('electron').ipcRenderer.send('get\_product\_code', this.state.products[idx].id);  } else {  this.setState({  barcode\_source\_number\_val: "",  barcode\_source\_number\_is\_valid: false,  barcode\_obj: null  });  }  }  get\_product\_input = () => {  return (  <>  <p style={{ marginTop: "20px", textAlign: "center", fontSize: "20px"}}>  Choose product from list:  </p>  <div className="justify-content-center mx-5" style={{marginBottom: "25px"}}>  <input style={{width: "500px"}} value={this.state.product\_name\_input} onChange={(e) => this.handle\_set\_product(e.currentTarget.value)} className="form-control mx-auto" list="datalistOptions1" id="exampleDataList" placeholder="Type to search..."/>  <datalist id="datalistOptions1">  {this.state.products && this.state.products.sort((x, y) => x.code - y.code).map(x =>(  <option key={x.id} value={x.name} itemID={x.id} />  ))}  </datalist>  {this.state.product\_current\_idx > 0 &&  <>  <p style={{ marginTop: "10px", textAlign: "center", fontSize: "20px"}}>Info about product:</p>  <div className="row g-4">  <div className="col-6">  <p><b>Product name</b>: {this.state.products[this.state.product\_current\_idx].name}</p>  </div>  <div className="col-3">  <p><b>Product type</b>: {this.state.products[this.state.product\_current\_idx].type}</p>  </div>  <div className="col-3">  <p><b>Country</b>: {this.state.countries.find(y => y.id === this.state.manufactures.find(x => x. id === this.state.products[this.state.product\_current\_idx].manufacture\_id).country\_id).name}</p>  </div>  </div>  <div className="row g-4">  <div className="col-6">  <p style={{marginBottom: "0"}}><b>Manufacture name</b>: {this.state.manufactures && this.state.manufactures.find(y => y.id === this.state.products[this.state.product\_current\_idx].manufacture\_id)?.name}</p>  </div>  <div className="col-2">  <p style={{marginBottom: "0"}}><b>Product code</b>: {this.state.products[this.state.product\_current\_idx].code.toString().padStart(5, "0")}</p>  </div>  <div className="col-2">  <p style={{marginBottom: "0"}}><b>Product price</b>: {this.state.products[this.state.product\_current\_idx].price}</p>  </div>  <div className="col-2">  <p style={{marginBottom: "0"}}><b>Product color</b>: <span style={{ color: this.state.products[this.state.product\_current\_idx].color, backgroundColor: this.state.products[this.state.product\_current\_idx].color}}>\_\_\_\_\_\_</span></p>  </div>  </div>  </>  }  </div>  </>  );  }  render() {  return (  <div style={{ marginTop: "20px" }}> {/\*height: "calc(100% - 20px)", \*/}  <p style={{ marginTop: "0", textAlign: "center", fontSize: "20px"}}>  You can generate <b>{BARCODE\_NAME}</b> barcode from:  </p>  <ul className="nav nav-tabs justify-content-center">  <li className="nav-item">  <a className={this.state.is\_product\_input ? "nav-link" : "nav-link active"} href="#" onClick={() => this.change\_page(false)}>Text</a>  </li>  <li className="nav-item">  <a className={!this.state.is\_product\_input ? "nav-link" : "nav-link active"} href="#" onClick={() => this.change\_page(true)}>Product</a>  </li>  </ul>  {this.state.is\_product\_input ? this.get\_product\_input() : this.get\_text\_input()}  {this.append\_results\_block\_if\_needed()}  </div>  );  }  }  export default Encode; |

Лістинг програмного модулю розпізнавання штрих-кодової позначки із вводу користувача з функцією зчитування штрих-коду з файлу:

|  |
| --- |
| app/decode.tsx |
| import React from "react";  import EAN13 from "./ean13";  // const { ipcRenderer } = require('electron');  import { IProduct } from "./../products";  import { IManufacture } from "./../manufacturers";  import { ICountry } from "./../countries";  type DecodeState = {  is\_image\_input: boolean,  text\_barcode\_digits: string,  png\_data: string | null,  products: IProduct[],  manufactures: IManufacture[],  countries: ICountry[],  product\_idx: number  };  class Decode extends React.Component {  state: DecodeState = {  is\_image\_input: false,  text\_barcode\_digits: "",  png\_data: null,  products: [],  manufactures: [],  countries: [],  product\_idx: -1  }  change\_page = (is\_image\_input: boolean) => {  if (is\_image\_input === this.state.is\_image\_input) {  return;  }  this.setState({is\_image\_input});  this.reset\_text\_input();  }  upload\_image\_file = () => {  const reader = new FileReader();  const blob = (document.getElementById('file\_image') as HTMLInputElement).files[0];  reader.readAsArrayBuffer(blob);  reader.onload = () => this.parse\_image\_file(reader.result);  const urlReader = new FileReader();  urlReader.readAsDataURL(blob);  urlReader.onload = () => this.setState({png\_data: urlReader.result});  }  parse\_image\_file = (payload: string | ArrayBuffer) => {  window.require('electron').ipcRenderer.on('png\_parse\_reply', (\_, arg) => {  this.setState({text\_barcode\_digits: arg})  this.getIdxOfProduct(arg);  });  window.require('electron').ipcRenderer.send('png\_parse', payload);  }  show\_loaded\_image = () => {  if (this.state.png\_data !== null) {  return (  <>  <p style={{ marginTop: "20px", fontSize: "18px", marginBottom: "3px"}}>Image loaded from file:</p>  <img src={this.state.png\_data}/>  </>  );  }  return (<></>);  }  get\_image\_input = () => {  return (  <>  <p style={{ marginTop: "20px", textAlign: "center", fontSize: "20px"}}>  Upload the image of barcode (.png) by clicking the button below:  </p>  <div style={{display: "flex", flexDirection: "column", alignItems: "center", marginBottom: "20px" }}>  <div style={{display: "flex"}}>  <button className="btn btn-secondary" style={{display: this.state.png\_data === null ? "none" : "block", marginRight: "10px"}} onClick={this.reset\_text\_input}>Reset uploaded image</button>  <button type="button" onClick={() => document.getElementById('file\_image').click()} style={{marginRight: "auto"}} className="btn btn-primary mx-auto"><i className="bi bi-upload"></i> Load barcode image from file (.png)</button>  </div>  <input id='file\_image' accept=".png" onChange={() => this.upload\_image\_file()} type='file' hidden/>  {this.show\_loaded\_image()}  </div>  </>  );  }  decodeBarcode = () => {  const result = EAN13.decode(this.state.text\_barcode\_digits);  return result;  }  getIdxOfProduct = (text\_barcode\_digits: string) => {  const barcode = EAN13.decode(text\_barcode\_digits);  if (text\_barcode\_digits.length === 95 && barcode !== null && barcode.length === 12) {  window.require('electron').ipcRenderer.on('get\_product\_by\_code\_reply', (\_, arg: string) => {  const product\_idx = this.state.products.findIndex(x => x.id === arg);  this.setState({  product\_idx  });  });  window.require('electron').ipcRenderer.send('get\_product\_by\_code', barcode);  }  }  upload\_text\_file = () => {  const reader = new FileReader();  const blob = (document.getElementById('file\_text') as HTMLInputElement).files[0];  reader.readAsText(blob);  reader.onload = () => {  this.setState({text\_barcode\_digits: reader.result.toString()});  this.getIdxOfProduct(reader.result.toString());  };    }  reset\_text\_input = () => {  this.setState({text\_barcode\_digits: "", png\_data: null, text\_barcode\_digits\_from\_png: null});  this.getIdxOfProduct("");  document.getElementById("textInput")?.focus();  if (!document.getElementById("textInput")) {  return;  }  (document.getElementById('file\_text') as HTMLInputElement).value = "";  }  tmp = (value: string) => {  this.setState({text\_barcode\_digits: value});  this.getIdxOfProduct(value);  }  get\_text\_input = () => {  return (  <>  <p style={{ marginTop: "20px", textAlign: "center", fontSize: "20px"}}>  Enter barcodes digits <i>(binary representation)</i> in input below:  </p>  <div style={{display: "flex"}}>  <textarea id="textInput" value={this.state.text\_barcode\_digits} maxLength={95} minLength={95} onChange={(e) => this.tmp(e.currentTarget.value)} style={{ fontFamily: "monospace", width: "920px", height: "30px", resize: "none", display: "block", marginLeft: "auto", marginRight: "5px" }}></textarea>  <button type="button" onClick={() => document.getElementById('file\_text').click()} style={{marginRight: "auto"}} className="btn btn-outline-success btn-sm mr-auto"><i className="bi bi-upload"></i> Load binary digits from file (.txt)</button>  <input id='file\_text' accept=".txt" onChange={() => this.upload\_text\_file()} type='file' hidden/>  </div>  <p className="source-barcode-actions" style={{marginTop: "20px"}}>  <button className="btn btn-secondary" onClick={this.reset\_text\_input}>Reset barcode input</button>  </p>  </>  );  }  results\_block = () => {  const res = this.decodeBarcode();  if (this.state.text\_barcode\_digits.length === 95 && res !== null && res.length === 12) {  return (  <div>  <hr style={{ margin: "0 30px" }}/>  <p style={{ textAlign: "center", marginTop: "10px", marginBottom: "3px", fontSize: "20px", fontWeight: 500 }}>  Results:  </p>  {this.state.text\_barcode\_digits !== null && this.state.is\_image\_input === true &&  <>  <p style={{textAlign: "center", marginBottom: "1px", fontSize: "17px", fontStyle: "italic"}}> Parsed binary digits from image:</p>  <p style={{fontFamily: "monospace", color: "black", textAlign: "center"}}>{this.state.text\_barcode\_digits}</p>  </>  }  <p style={{ textAlign: "center", marginTop: "0.8rem", marginBottom: "0"}}><b>Source barcode you can see below</b></p>  <pre className="source-barcode" style={{fontSize: "50px", marginTop: "8px", lineHeight: "40px", overflow: "hidden"}}>  <span><b>{res}</b></span>  </pre>  {this.state.product\_idx > 0 &&  <>  <p style={{ marginTop: "8px", marginBottom: "4px", textAlign: "center", fontSize: "16px"}}>Info about product:</p>  <div className="mx-auto" style={{width: "400px"}}>  <p style={{marginBottom: "3px", fontSize: "15px"}}><b>Product name</b>: {this.state.products[this.state.product\_idx].name}</p>  <p style={{marginBottom: "3px", fontSize: "15px"}}><b>Product type</b>: {this.state.products[this.state.product\_idx].type}</p>  <p style={{marginBottom: "3px", fontSize: "15px"}}><b>Country</b>: {this.state.countries.find(y => y.id === this.state.manufactures.find(x => x. id === this.state.products[this.state.product\_idx].manufacture\_id).country\_id).name}</p>  <p style={{marginBottom: "3px", fontSize: "15px"}}><b>Manufacture name</b>: {this.state.manufactures && this.state.manufactures.find(y => y.id === this.state.products[this.state.product\_idx].manufacture\_id)?.name}</p>  <p style={{marginBottom: "3px", fontSize: "15px"}}><b>Product code</b>: {this.state.products[this.state.product\_idx].code.toString().padStart(5, "0")}</p>  <p style={{marginBottom: "3px", fontSize: "15px"}}><b>Product price</b>: {this.state.products[this.state.product\_idx].price}</p>  <p style={{marginBottom: "0px", fontSize: "15px"}}><b>Product color</b>: <span style={{ color: this.state.products[this.state.product\_idx].color, backgroundColor: this.state.products[this.state.product\_idx].color}}>\_\_\_\_\_\_</span></p>  </div>  </>  }  </div>  );  } else if (this.state.text\_barcode\_digits.length === 0) {  return (  <div style={{ display: "flex", flexDirection: "column", alignItems: "center", justifyContent: "center", alignContent: "center" }}>  <p style={{ textAlign: "center", fontSize: "20px" }}>No results...</p>  <p style={{ textAlign: "center", marginTop: "0px", fontStyle: "italic", color: "GrayText" }}>  Please, {this.state.is\_image\_input ? " upload image of barcode " : " enter digits in the input "} for the purpose of getting here results – source of barcode  </p>  </div>  );  } else if (this.state.text\_barcode\_digits.length === 95 && res === null) {  return (  <div className="callout callout-error mx-auto" style={{width: "400px"}}>  <h5>Couldn't validate barcodes digits</h5>  <p style={{marginBottom: "0"}}>There is an error in your input, check it please</p>  </div>  );  } else {  return (  <div className="callout callout-warning mx-auto" style={{width: "400px"}}>  <h5>Your input is not complete</h5>  <p style={{marginBottom: "0"}}>You need to paste {95 - this.state.text\_barcode\_digits.length} digits more</p>  </div>  );  }  }  componentDidMount() {  if (!this.state.is\_image\_input) {  document.getElementById("textInput")?.focus();  }  window.require('electron').ipcRenderer.on('get\_all\_products\_reply', (\_, arg: DecodeState["products"]) => {  this.setState({  products: arg  });  });  window.require('electron').ipcRenderer.send('get\_all\_products');  window.require('electron').ipcRenderer.on('get\_all\_manufacturers\_reply', (\_, arg: DecodeState["manufactures"]) => {  this.setState({  manufactures: arg  });  });  window.require('electron').ipcRenderer.send('get\_all\_manufacturers');  window.require('electron').ipcRenderer.on('get\_all\_countries\_reply', (\_, arg: ICountry[]) => {  this.setState({  countries: arg  });  });  window.require('electron').ipcRenderer.send('get\_all\_countries');  }  render() {  return (  <div style={{ marginTop: "20px" }}> {/\*height: "calc(100% - 20px)", \*/}  <p style={{ marginTop: "0", textAlign: "center", fontSize: "20px"}}>  You can decode <b>EAN-13</b> barcode from:  </p>  <ul className="nav nav-tabs justify-content-center">  <li className="nav-item">  <a className={this.state.is\_image\_input ? "nav-link" : "nav-link active"} href="#" onClick={() => this.change\_page(false)}>Text</a>  </li>  <li className="nav-item">  <a className={!this.state.is\_image\_input ? "nav-link" : "nav-link active"} href="#" onClick={() => this.change\_page(true)}>Image</a>  </li>  </ul>  {this.state.is\_image\_input ? this.get\_image\_input() : this.get\_text\_input()}  {this.results\_block()}  </div>  );  }  }  export default Decode; |

Лістинг програмного модулю для управління сутностями (країни, виробники, продукти) з бази даних:

|  |
| --- |
| app/products.tsx |
| import React from "react";  import { ICountry } from "./../countries";  import { IManufacture } from "./../manufacturers";  import { IProduct } from "./../products";  type ProductsState = {  current\_page: "countries" | "manufacturers" | "products",  countries: ICountry[] | null,  manufacturers: (IManufacture)[] | null,  products: (IProduct)[] | null,  country\_name\_input: string,  country\_name\_input\_is\_valid: boolean,  country\_code\_input: string,  manufacturer\_code\_input: string,  manufacturer\_name\_input: string,  manufacturer\_id\_edit: string | null;  manufacture\_name\_input: string,  product\_name\_input: string,  product\_type\_input: string,  product\_color\_input: string,  product\_price\_input: string,  product\_code\_input: string,  product\_id\_edit: string | null;  };  class Products extends React.Component {  state: ProductsState = {  current\_page: "countries",  countries: [],  manufacturers: [],  products: [],  country\_name\_input: "",  country\_name\_input\_is\_valid: false,  country\_code\_input: "",  manufacturer\_code\_input: "",  manufacturer\_name\_input: "",  manufacturer\_id\_edit: null,  manufacture\_name\_input: "",  product\_name\_input: "",  product\_type\_input: "",  product\_color\_input: "#000000",  product\_price\_input: "",  product\_code\_input: "",  product\_id\_edit: null,  }  componentDidMount() {  this.change\_page("countries", true);  }  load\_mabufactures = () => {  window.require('electron').ipcRenderer.on('get\_all\_manufacturers\_reply', (\_, arg: ProductsState["manufacturers"]) => {  this.setState({  manufacturers: arg  });  });  window.require('electron').ipcRenderer.send('get\_all\_manufacturers');  }  load\_products = () => {  window.require('electron').ipcRenderer.on('get\_all\_products\_reply', (\_, arg: ProductsState["products"]) => {  this.setState({  products: arg  });  });  window.require('electron').ipcRenderer.send('get\_all\_products');  }  change\_page = (page: ProductsState["current\_page"], onLoad: boolean = false) => {  if (page != this.state.current\_page || onLoad) {  this.setState({  current\_page: page,  manufacturer\_id\_edit: null  });  if (page === "countries" || page === "manufacturers" || page === "products") {  window.require('electron').ipcRenderer.on('get\_all\_countries\_reply', (\_, arg: ICountry[]) => {  this.setState({  countries: arg  });  });  window.require('electron').ipcRenderer.send('get\_all\_countries');  }  if (page === "manufacturers" || page === "products") {  this.load\_mabufactures();  }  if (page === "products") {  this.load\_products();  }  }  }  country\_name\_input\_handler = (e: React.ChangeEvent<HTMLInputElement>) => {  const cur\_value = e.currentTarget.value;  const find\_val = this.state.countries.find(x => x.name === cur\_value);  if (find\_val) {  this.setState({  country\_name\_input: cur\_value,  country\_name\_input\_is\_valid: true,  });  (document.getElementById('exampleInputPassword1') as HTMLInputElement).setAttribute("min", find\_val.code\_start.toString());  const is\_end = !!find\_val.code\_end;  if (is\_end) {  (document.getElementById('exampleInputPassword1') as HTMLInputElement).setAttribute("max", find\_val.code\_end.toString());  this.setState({  country\_code\_input: find\_val.code\_start.toString(),  });  } else {  (document.getElementById('exampleInputPassword1') as HTMLInputElement).setAttribute("max", find\_val.code\_start.toString());  this.setState({  country\_code\_input: find\_val.code\_start.toString(),  });  }  } else {  this.setState({  country\_name\_input: cur\_value,  country\_name\_input\_is\_valid: false,  });  this.setState({  country\_code\_input: "",  });  }    }  add\_manufacturer\_to\_db = () => {  if (this.state.manufacturer\_id\_edit === null) {  window.require('electron').ipcRenderer.send('add\_manufacture', {  country\_id: this.state.countries.find(x => x.name === this.state.country\_name\_input).id,  country\_code: this.state.country\_code\_input,  code: this.state.manufacturer\_code\_input,  name: this.state.manufacturer\_name\_input,  description: ""  });  } else {  window.require('electron').ipcRenderer.send('update\_manufacture', {  id: this.state.manufacturer\_id\_edit,  country\_id: this.state.countries.find(x => x.name === this.state.country\_name\_input).id,  country\_code: this.state.country\_code\_input,  code: this.state.manufacturer\_code\_input,  name: this.state.manufacturer\_name\_input,  description: ""  });  }  }  add\_manufacturer\_click\_handler = () => {  this.add\_manufacturer\_to\_db();  this.setState({  country\_name\_input: "",  country\_name\_input\_is\_valid: false,  country\_code\_input: "",  manufacturer\_code\_input: "",  manufacturer\_name\_input: "",  manufacturer\_id\_edit: null  });  setTimeout(() => this.load\_mabufactures(), 400);  }  add\_product\_to\_db = () => {  if (this.state.product\_id\_edit === null) {  window.require('electron').ipcRenderer.send('add\_product', {  manufacture\_id: this.state.manufacturers.find(x => x.name === this.state.manufacture\_name\_input).id,  code: this.state.product\_code\_input,  name: this.state.product\_name\_input,  type: this.state.product\_type\_input,  color: this.state.product\_color\_input,  price: this.state.product\_price\_input,  });  } else {  window.require('electron').ipcRenderer.send('update\_product', {  id: this.state.product\_id\_edit,  manufacture\_id: this.state.manufacturers.find(x => x.name === this.state.manufacture\_name\_input).id,  code: this.state.product\_code\_input,  name: this.state.product\_name\_input,  type: this.state.product\_type\_input,  color: this.state.product\_color\_input,  price: this.state.product\_price\_input  });  }  }  add\_product\_click\_handler = () => {  this.add\_product\_to\_db();  this.setState({  manufacture\_name\_input: "",  product\_name\_input: "",  product\_type\_input: "",  product\_color\_input:"#000000",  product\_price\_input: "",  product\_code\_input: "",  product\_id\_edit: null,  });  setTimeout(() => this.load\_products(), 400);  }  delete\_manufacturer = (id: string) => {  window.require('electron').ipcRenderer.send('delete\_manufacture', id);  setTimeout(() => this.load\_mabufactures(), 400);  }  delete\_product = (id: string) => {  window.require('electron').ipcRenderer.send('delete\_product', id);  setTimeout(() => this.load\_products(), 400);  }  set\_manufacturer\_to\_edit = (id: string) => {  const cur\_m = this.state.manufacturers.find(x => x.id === id);  const countr = this.state.countries.find(x => x.id === cur\_m.country\_id);  this.setState({  manufacturer\_id\_edit: id,  country\_name\_input: countr.name,  country\_name\_input\_is\_valid: true,  country\_code\_input: cur\_m.country\_code.toString(),  manufacturer\_code\_input: cur\_m.code.toString(),  manufacturer\_name\_input: cur\_m.name,  });  }  set\_product\_to\_edit = (id: string) => {  const cur\_p = this.state.products.find(x => x.id === id);  const manuf = this.state.manufacturers.find(x => x.id === cur\_p.manufacture\_id);  this.setState({  manufacture\_name\_input: manuf.name,  product\_name\_input: cur\_p.name,  product\_type\_input: cur\_p.type,  product\_color\_input: cur\_p.color,  product\_price\_input: cur\_p.price.toString(),  product\_code\_input: cur\_p.code.toString(),  product\_id\_edit: id,  });  }  cancel\_product\_to\_edit = () => {  this.setState({  manufacture\_name\_input: "",  product\_name\_input: "",  product\_type\_input: "",  product\_color\_input: "#000000",  product\_price\_input: "",  product\_code\_input: "",  product\_id\_edit: null,  });  }  cancel\_manufacturer\_to\_edit = () => {  this.setState({  manufacturer\_id\_edit: null,  country\_name\_input: "",  country\_name\_input\_is\_valid: false,  country\_code\_input: "",  manufacturer\_code\_input: "",  manufacturer\_name\_input: "",  });  }  render\_products\_page = () => {  const is\_add\_product\_button\_active = !!this.state.manufacturers && !!this.state.manufacturers.find(x => x.name === this.state.manufacture\_name\_input) &&  this.state.product\_code\_input && this.state.product\_name\_input.length > 2 && this.state.product\_type\_input.length > 2 &&  this.state.product\_color\_input.length > 2 && this.state.product\_price\_input.length > 0 &&  Number.isInteger(+this.state.product\_code\_input) && parseFloat(this.state.product\_price\_input) > 0;  return (  <div>  <p style={{textAlign: "center", fontSize: "25px", marginTop: "10px", marginBottom: "0"}}>List of products</p>  <div style={{width: "1417px", height: "300px", maxWidth: "1417px"}} className="table-responsive mx-auto">  <table style={{width: "1400px"}} className="table table-striped table-hover">  <thead className="sticky-border manuf" style={{position: "sticky", top: "0", backgroundColor: "#fff" }}>  <tr>  <th style={{width: "120px"}}>Code</th>  <th>Manufacture name</th>  <th>Product type</th>  <th>Product name</th>  <th>Price</th>  <th>Color</th>  <th style={{width: "85px"}}>Actions</th>  </tr>  </thead>  <tbody>  {this.state.products && this.state.products.sort((x, y) => x.code - y.code).map(x => (  <tr key={x.id}>  <td>{x.code.toString().padStart(5, "0")}</td>  <td>{this.state.manufacturers && this.state.manufacturers.find(y => y.id === x.manufacture\_id)?.name}</td>  <td>{x.type}</td>  <td>{x.name}</td>  <td>{x.price}</td>  <td><span style={{ color: x.color, backgroundColor: x.color}}>\_\_\_\_\_\_</span></td>  <td><i style={{marginRight: "15px", cursor: "pointer"}} id={"e\_p" + x.id} onClick={(e) => this.set\_product\_to\_edit(e.currentTarget.id.substr(3))} className="bi bi-pencil"></i><i style={{cursor: "pointer"}} id={"d\_p" + x.id} onClick={(e) => this.delete\_product(e.currentTarget.id.substr(3))} className="bi bi-trash"></i></td>  </tr>  ))}  </tbody>  </table>  </div>  <div className="mx-auto" style={{ marginTop: "15px", width: "700px" }}>  <p style={{textAlign: "center", fontSize: "25px",}}>{!this.state.manufacturer\_id\_edit ? "Add new product:" : "Edit product:"}</p>  <form id="add\_product">  <div className="row g-3 mb-3">  <div className="col-7">  <label htmlFor="exampleDataList" className="form-label">Manufacture name</label>  <input value={this.state.manufacture\_name\_input} onChange={(e) => this.setState({manufacture\_name\_input: e.currentTarget.value})} className="form-control" list="datalistOptions1" id="exampleDataList" placeholder="Type to search..."/>  <datalist id="datalistOptions1">  {this.state.manufacturers && this.state.manufacturers.sort((x, y) => x.code - y.code).map(x =>(  <option key={x.id} value={x.name} itemID={x.id} />  ))}  </datalist>  </div>  <div className="col-5">  <label htmlFor="exampleInputPassword1" className="form-label">Product type</label>  <input type="text" value={this.state.product\_type\_input} onChange={(e) => this.setState({product\_type\_input: e.currentTarget.value})} className="form-control" id="exampleInputPassword1"/>  </div>  </div>  <div className="row g-3 mb-3">  <div className="col-md">  <label htmlFor="exampleInputPassword1" className="form-label">Product name</label>  <input type="text" value={this.state.product\_name\_input} onChange={(e) => this.setState({product\_name\_input: e.currentTarget.value})} className="form-control" id="exampleInputPassword1"/>  </div>  </div>  <div className="row g-3 mb-3">  <div className="col-2">  <label htmlFor="exampleInputPassword1" className="form-label">Color</label>  <input type="color" style={{height: "38px"}} value={this.state.product\_color\_input} onChange={(e) => this.setState({product\_color\_input: e.currentTarget.value})} className="form-control" id="exampleInputPassword1"/>  </div>  <div className="col-5">  <label htmlFor="exampleInputPassword1" className="form-label">Price</label>  <input type="number" step="0.01" min={0} value={this.state.product\_price\_input} onChange={(e) => this.setState({product\_price\_input: e.currentTarget.value})} className="form-control" id="exampleInputPassword1"/>  </div>  <div className="col-5">  <label htmlFor="exampleInputPassword1" className="form-label">Product code</label>  <input type="number" min={0} step={0} max={99999} value={this.state.product\_code\_input} onChange={(e) => this.setState({product\_code\_input: e.currentTarget.value.substr(0, 5)})} className="form-control" id="exampleInputPassword1"/>  </div>  </div>  <button disabled={!is\_add\_product\_button\_active} onClick={() => this.add\_product\_click\_handler()} type="button" className="btn btn-primary">{!this.state.product\_id\_edit ? "Add" : "Update"}</button>  <button style={{ marginLeft: "20px", display: !this.state.product\_id\_edit ? "none" : ""}} onClick={() => this.cancel\_product\_to\_edit()} type="button" className="btn btn-secondary">Cancel</button>  </form>  </div>  </div>  );  }  render\_manufacturers\_page = () => {  const is\_add\_manufacturer\_button\_active = !!this.state.countries.find(x => x.name === this.state.country\_name\_input) &&  this.state.country\_name\_input\_is\_valid && this.state.manufacturer\_code\_input.length === 4 &&  this.state.manufacturer\_name\_input.length > 2 && Number.isInteger(+this.state.manufacturer\_code\_input);  return (  <div>  <p style={{textAlign: "center", fontSize: "25px", marginTop: "10px", marginBottom: "0"}}>List of manufacturers</p>  <div style={{width: "1417px", height: "300px", maxWidth: "1417px"}} className="table-responsive mx-auto">  <table style={{width: "1400px"}} className="table table-striped table-hover">  <thead className="sticky-border manuf" style={{position: "sticky", top: "0", backgroundColor: "#fff" }}>  <tr>  <th style={{width: "120px"}}>Country code</th>  <th style={{width: "160px"}}>Manufacture code</th>  <th style={{width: "360px"}}>Country name</th>  <th>Manufacture name</th>  <th style={{width: "85px"}}>Actions</th>  </tr>  </thead>  <tbody>  {this.state.manufacturers && this.state.manufacturers.sort((x, y) => x.code - y.code).map(x => (  <tr key={x.id}>  <td>{x.country\_code.toString().padStart(3, "0")}</td>  <td>{x.code.toString().padStart(4, "0")}</td>  <td>{this.state.countries.find(y => y.id === x.country\_id)?.name}</td>  <td>{x.name}</td>  <td><i style={{marginRight: "15px", cursor: "pointer"}} id={"e\_m" + x.id} onClick={(e) => this.set\_manufacturer\_to\_edit(e.currentTarget.id.substr(3))} className="bi bi-pencil"></i><i style={{cursor: "pointer"}} id={"d\_m" + x.id} onClick={(e) => this.delete\_manufacturer(e.currentTarget.id.substr(3))} className="bi bi-trash"></i></td>  </tr>  ))}  </tbody>  </table>  </div>  <div className="mx-auto" style={{ marginTop: "15px", width: "700px" }}>  <p style={{textAlign: "center", fontSize: "25px",}}>{!this.state.manufacturer\_id\_edit ? "Add new manufacture:" : "Edit manufacture:"}</p>  <form id="add\_manufacturer">  <div className="row g-2 mb-3">  <div className="col-md">  <label htmlFor="exampleDataList" className="form-label">Country name</label>  <input value={this.state.country\_name\_input} onChange={this.country\_name\_input\_handler} className="form-control" list="datalistOptions" id="exampleDataList" placeholder="Type to search..."/>  <datalist id="datalistOptions">  {this.state.countries && this.state.countries.sort((x, y) => x.code\_start - y.code\_start).map(x =>(  <option key={x.id} value={x.name} itemID={x.id} />  ))}  </datalist>  </div>  <div className="col-md">  <label htmlFor="exampleInputPassword1" className="form-label">Country code</label>  <input type="number" value={this.state.country\_code\_input} onChange={(e) => this.setState({country\_code\_input: e.currentTarget.value})} className="form-control" id="exampleInputPassword1"/>  </div>  </div>  <div className="mb-3">  <label htmlFor="exampleInputEmail1" className="form-label">Manafacture code</label>  <input type="text" pattern="\d\*" value={this.state.manufacturer\_code\_input} maxLength={4} onChange={(e) => this.setState({manufacturer\_code\_input: e.currentTarget.value})} className="form-control" id="exampleInputEmail1"/>  </div>  <div className="mb-3">  <label htmlFor="exampleInputEmail1" className="form-label">Manafacture name</label>  <input type="text" value={this.state.manufacturer\_name\_input} onChange={(e) => this.setState({manufacturer\_name\_input: e.currentTarget.value})} className="form-control" id="exampleInputEmail1"/>  </div>  <button disabled={!is\_add\_manufacturer\_button\_active} onClick={() => this.add\_manufacturer\_click\_handler()} type="button" className="btn btn-primary">{!this.state.manufacturer\_id\_edit ? "Add" : "Update"}</button>  <button style={{ marginLeft: "20px", display: !this.state.manufacturer\_id\_edit ? "none" : ""}} onClick={() => this.cancel\_manufacturer\_to\_edit()} type="button" className="btn btn-secondary">Cancel</button>  </form>  </div>  </div>  );  }  render\_countries\_page = () => {  return (  <div>  <p style={{textAlign: "center", fontSize: "25px", marginTop: "10px", marginBottom: "0"}}>List of countries</p>  <div style={{width: "618px", maxHeight: "618px", maxWidth: "618px"}} className="table-responsive mx-auto">  <table style={{width: "600px"}} className="table table-striped table-hover">  <thead className="sticky-border" style={{position: "sticky", top: "0", backgroundColor: "#fff" }}>  <tr>  <th>Codes</th>  <th>Country name</th>  </tr>  </thead>  <tbody>  {this.state.countries && this.state.countries.sort((x, y) => x.code\_start - y.code\_start).map(x => (  <tr key={x.id}>  <td>{x.code\_end ? `${x.code\_start.toString().padStart(3, "0")} — ${x.code\_end}` : x.code\_start}</td>  <td>{x.name}</td>  </tr>  ))}  </tbody>  </table>  </div>  </div>  );  }  render() {  return (  <div style={{ marginTop: "20px" }}>  <ul className="nav nav-tabs justify-content-center">  <li className="nav-item">  <a className={this.state.current\_page !== "countries" ? "nav-link" : "nav-link active"} href="#" onClick={() => this.change\_page("countries")}>Countries</a>  </li>  <li className="nav-item">  <a className={this.state.current\_page !== "manufacturers" ? "nav-link" : "nav-link active"} href="#" onClick={() => this.change\_page("manufacturers")}>Manufacturers</a>  </li>  <li className="nav-item">  <a className={this.state.current\_page !== "products" ? "nav-link" : "nav-link active"} href="#" onClick={() => this.change\_page("products")}>Prodcuts</a>  </li>  </ul>  {this.state.current\_page === "countries" ?  this.render\_countries\_page() : this.state.current\_page === "manufacturers" ?  this.render\_manufacturers\_page() : this.render\_products\_page()  }  </div>  );  }  }  export default Products; |

Лістинг програмного модулю для відображення основної сторінки:

|  |
| --- |
| app/app.tsx |
| import React from "react";  import Decode from "./decode";  import Products from "./products";  import Encode from "./encode";  type AppState = { current\_page: "encode" | "decode" | "products" };  class App extends React.Component {  state: AppState = {  current\_page: "encode"  }  render() {  const current\_page = this.state.current\_page;  return (  <div > {/\* style={{ height: "calc(100% - 51px)" \*/}  <div style={{ paddingTop: "15px", paddingBottom: "15px", backgroundColor: "#fff6e4", borderBottom: "1px solid #111d38" }}>  <nav className="nav nav-pills nav-fill justify-content-center mx-auto" style={{ width: "500px" }}>  <a className={current\_page !== "products" ? "nav-link" : "nav-link active"} href="#" onClick={() => this.setState({current\_page: "products"})}>Products</a>  <a className={current\_page !== "encode" ? "nav-link" : "nav-link active"} href="#" onClick={() => this.setState({current\_page: "encode"})}>Encode</a>  <a className={current\_page !== "decode" ? "nav-link" : "nav-link active"} href="#" onClick={() => this.setState({current\_page: "decode"})}>Decode</a>  </nav>  </div>  {current\_page == "encode" ? <Encode/> : current\_page == "products" ? <Products/> : <Decode/>}  </div>  );  }  }  export default App; |

Лістинг програмного модулю форми вводу штрих-коду:

|  |
| --- |
| app/digits.tsx |
| import React from "react";  import useDigitInput, { InputAttributes } from 'react-digit-input';  function getArrayOfDigitInputs(count: number, digits: InputAttributes[]) {  let result = [  <input id="first-input-digit" className="input-digit" key={0} inputMode="decimal" autoFocus {...digits[0]} />  ];  for (let i = 1; i < count; i++) {  if (i == count - 1) {  result.push(<input className="input-digit" key={i} id="last-input-digit" inputMode="decimal" {...digits[i]} />);  } else {  result.push(<input className="input-digit" key={i} inputMode="decimal" {...digits[i]} />);  }  }  return result;  }  export default function DigitsPanel(props: { count\_digits: number, on\_value\_change: (value: string) => void, value?: string }) {  const [value, onChange] = React.useState(props.value ?? '');  if (props.value !== value) {  onChange(props.value);  }  const digits = useDigitInput({  acceptedCharacters: /^[0-9]$/,  length: props.count\_digits,  value,  onChange: x => {  onChange(x);  props.on\_value\_change(x);  },  });  return (  <div className="input-group">  {getArrayOfDigitInputs(props.count\_digits, digits)}  </div>  );  } |

Лістинг моделі, яка відображає сутність «Держави»:

|  |
| --- |
| countries.ts |
| import { Schema, Document } from "mongoose";  import mongoose from "mongoose";  export interface ICountry extends Document {  code\_start: number;  code\_end?: number;  name: string;  }  const CountrySchema: Schema = new Schema({  code\_start: { type: Number, required: true, unique: true },  code\_end: { type: Number, required: false },  name: { type: String, required: true },  });  const CountryModel = mongoose.model<ICountry>("Country", CountrySchema);  export default CountryModel; |

Лістинг моделі, яка відображає сутність «Виробники»:

|  |
| --- |
| manufacturers.ts |
| import { Schema, Document } from "mongoose";  import mongoose from "mongoose";  export interface IManufacture extends Document {  country\_id: mongoose.Types.ObjectId;  country\_code: number;  code: number;  name: string;  description: string;  }  const ManufactureSchema: Schema = new Schema({  country\_id: { type: mongoose.Schema.Types.ObjectId, required: true, ref: "Country" },  country\_code: { type: Number, required: true },  code: { type: Number, required: true },  name: { type: String, required: true },  description: { type: String, required: false, default: "" }  });  const ManufactureModel = mongoose.model<IManufacture>("Manufacture", ManufactureSchema);  export default ManufactureModel; |

Лістинг моделі, яка відображає сутність «Товари»:

|  |
| --- |
| products.ts |
| import { Schema, Document } from "mongoose";  import mongoose from "mongoose";  export interface IProduct extends Document {  manufacture\_id: mongoose.Types.ObjectId;  code: number;  name: string;  type: string;  color: string;  price: number;  }  const ProductSchema: Schema = new Schema({  manufacture\_id: { type: mongoose.Schema.Types.ObjectId, required: true, ref: "Manufacture" },  code: { type: Number, required: true },  name: { type: String, required: true },  type: { type: String, required: true },  color: { type: String, required: true },  price: { type: Number, required: true }  });  const ProductModel = mongoose.model<IProduct>("Product", ProductSchema);  export default ProductModel; |

Лістинг програмного модуля сервера:

|  |
| --- |
| main.ts |
| import { app, BrowserWindow, ipcMain } from "electron";  import \* as path from "path";  import \* as dotenv from 'dotenv';  import mongoose from "mongoose";  import { PNG } from 'pngjs';  import CountryModel, { ICountry } from "./countries";  import ManufactureModel, { IManufacture } from "./manufacturers";  import ProductModel, { IProduct } from "./products";  dotenv.config();  function createWindow() {  // Create the browser window.  const mainWindow = new BrowserWindow({  height: 1000,  width: 1500,  minHeight: 1000,  minWidth: 1500,  autoHideMenuBar: true, // set to true  webPreferences: {  nodeIntegration: true,  webSecurity: false,  contextIsolation: false,  },  });  mainWindow.menuBarVisible = false;  if (process.env.NODE\_ENV === "development") {  mainWindow.loadURL(`http://localhost:4000`);  mainWindow.webContents.openDevTools();  } else {  mainWindow.loadFile(path.join(\_\_dirname, "../index.html"))  }  // and load the index.html of the app.  // mainWindow.loadFile(path.join(\_\_dirname, "../index.html"));  // Open the DevTools.  // mainWindow.webContents.openDevTools();  }  // This method will be called when Electron has finished  // initialization and is ready to create browser windows.  // Some APIs can only be used after this event occurs.  app.on("ready", () => {  mongoose.connect(process.env.DB\_URL)  .catch(err => err ? console.error(err) : console.log("Opened connection with db"))  .then(() => createWindow())  .catch(err => console.error(err));  app.on("activate", function () {  // On macOS it's common to re-create a window in the app when the  // dock icon is clicked and there are no other windows open.  if (BrowserWindow.getAllWindows().length === 0) {  createWindow();  }  });  });  // Quit when all windows are closed, except on macOS. There, it's common  // for applications and their menu bar to stay active until the user quits  // explicitly with Cmd + Q.  app.on("window-all-closed", () => {  if (process.platform !== "darwin") {  app.quit();  }  });  // In this file you can include the rest of your app"s specific main process  // code. You can also put them in separate files and require them here.  //  ipcMain.on('get\_all\_countries', function (event) {  CountryModel.find({}).then((data) => {  event.reply('get\_all\_countries\_reply', data.map(x => {  return {  name: x.name,  code\_start: x.code\_start,  code\_end: x.code\_end,  id: x.id  }  }));  }).catch((e) => {  console.log(e);  });  });  ipcMain.on('get\_all\_manufacturers', function (event) {  ManufactureModel.find({}).then((data) => {  event.reply('get\_all\_manufacturers\_reply', data.map(x => {  return {  name: x.name,  code: x.code,  country\_id: x.country\_id.toString(),  country\_code: x.country\_code,  id: x.id,  }  }));  }).catch((e) => {  console.log(e);  });  });  ipcMain.on('add\_manufacture', function (\_, arg: IManufacture) {  new ManufactureModel({  country\_id: arg.country\_id,  country\_code: arg.country\_code,  code: arg.code,  name: arg.name,  description: arg.description  }).save().then((data => {  // console.log(data);  }));  });  ipcMain.on('update\_manufacture', function (\_, arg: IManufacture) {  ManufactureModel.findByIdAndUpdate(arg.id, {  $set: {  country\_id: arg.country\_id,  country\_code: arg.country\_code,  code: arg.code,  name: arg.name  }  }).then(data => {  }).catch(e => {  console.log(e);  });  });  ipcMain.on('delete\_manufacture', function (\_, arg: string) {  ManufactureModel.findByIdAndRemove(arg).then(data => {  });  });  ipcMain.on('get\_all\_products', function (event) {  ProductModel.find({}).then((data) => {  event.reply('get\_all\_products\_reply', data.map(x => {  return {  manufacture\_id: x.manufacture\_id.toString(),  code: x.code,  name: x.name,  type: x.type,  color: x.color,  price: x.price,  id: x.id,  }  }));  }).catch((e) => {  console.log(e);  });  });  ipcMain.on('add\_product', function (\_, arg: IProduct) {  new ProductModel({  manufacture\_id: arg.manufacture\_id,  code: arg.code,  name: arg.name,  type: arg.type,  color: arg.color,  price: arg.price,  }).save().then((data => {  // console.log(data);  }));  });  ipcMain.on('update\_product', function (\_, arg: IProduct) {  ProductModel.findByIdAndUpdate(arg.id, {  $set: {  manufacture\_id: arg.manufacture\_id,  code: arg.code,  name: arg.name,  type: arg.type,  color: arg.color,  price: arg.price,  }  }).then(data => {  }).catch(e => {  console.log(e);  });  });  ipcMain.on('delete\_product', function (\_, arg: string) {  ProductModel.findByIdAndRemove(arg).then(data => {  });  });  ipcMain.on('get\_product\_by\_code', function (event, arg: string) {  if (arg && arg.length == 12) {  const c\_code = arg.substr(0, 3);  const m\_code = arg.substr(3, 4);  const p\_code = arg.substr(7);  ManufactureModel.findOne({  country\_code: +c\_code,  code: +m\_code  }).then(data => {  console.log(data);  ProductModel.findOne({  manufacture\_id: data.id,  code: +p\_code  }).then(res => {  event.reply('get\_product\_by\_code\_reply', res.id);  });  })  }  });  ipcMain.on('get\_product\_code', function (event, arg: string) {  let p\_code: string;  let m\_code: string;  let c\_code: string;  ProductModel.findById(arg).then(data => {  p\_code = data.code.toString().padStart(5, "0");  ManufactureModel.findById(data.manufacture\_id).then(man => {  m\_code = man.code.toString().padStart(4, "0")  c\_code = man.country\_code.toString().padStart(3, "0");  event.reply('get\_product\_code\_reply', c\_code + m\_code + p\_code);  });  });  });  //  ipcMain.on('png\_parse', (event, arg) => {  const data = arg as ArrayBuffer;  new PNG({ filterType: 4 }).parse(Buffer.from(data)).on("parsed", function () {  const arr\_data\_raw: number[] = [];  for (let y = 50; y == 50; y++) {  for (let x = 0; x < this.width; x++) {  const idx = (this.width \* y + x) << 2;  const red = this.data[idx];  arr\_data\_raw.push(red === 255 ? 0 : 1);  }  }  arr\_data\_raw.splice(0, 14);  const shrinked\_arr\_data\_raw: number[] = [];  for (let i = 0; i < arr\_data\_raw.length; i += 2) {  shrinked\_arr\_data\_raw.push(arr\_data\_raw[i]);  }  shrinked\_arr\_data\_raw.splice(95);  event.reply('png\_parse\_reply', shrinked\_arr\_data\_raw.join(""));  });  }); |

**Демонстрація програмного забезпечення**

Графічний інтерфейс для кодування штрих-кодової позначки із вводу користувача:

Graphical user interface, text, application, email

Description automatically generated

Результат кодування штрих-кодової позначки із вводу користувача:

Graphical user interface, text, email

Description automatically generated

Процес збереження зображення штрих-коду у файл:

Graphical user interface, text

Description automatically generated

Процес збереження бінарної інформації штрих-коду в текстовий файл:

Graphical user interface, text

Description automatically generated

Відображення інформації про держави та їх коди у БД:

Graphical user interface

Description automatically generated

Відображення інформації про виробників та їх коди у БД:

Graphical user interface, application

Description automatically generated

Додавання нового виробників у БД:

Graphical user interface, text, application, email

Description automatically generated

Редагування інформації про виробника у БД:

Graphical user interface, text, application, email

Description automatically generated

Відображення інформації про продукти та їх характеристики у БД:

Graphical user interface, application

Description automatically generated

Додавання продукту у БД:

Graphical user interface, application, website

Description automatically generated

Редагування інформації про продукт у БД:

Graphical user interface, application

Description automatically generated

Процес створення штрих-кодової позначки з товару у БД:

Graphical user interface, text, application, email

Description automatically generated

Декодування штрих-коду з бінарної інформації та відображення товару, який зареєстрований за даним штрих-кодом:

Graphical user interface, text, application, email

Description automatically generated

Декодування штрих-коду із зображення та відображення інформації про товар, який зареєстрований за даним штрих-кодом:

Graphical user interface, text, application

Description automatically generated

**Висновок**

Під час вивчення дисципліни було виконано цикл лабораторних робіт. В результаті циклу лабораторних робіт було створено пакет програм, що включають програму кодування вихідної інформації, програму друкування штрихової позначки, програму декодування зчитаної штрихової позначки, а також роботу з БД, де зберігається інформація про товари і які кодуються штрих-кодовими позначками.