

Практическая работа № 4

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА

Цель работы: программная реализация компьютерного тренажера

Методические указания

В отчете следует представить фрагменты программного кода с комментариями, поясняющими программную реализацию приложения. В отчет добавляем скриншот с доски проекта из Kaiten с текущим уровнем работы над проектом.

Отчет содержит: титульный лист, цель работы, порядок выполнения работы, вывод и список использованных источников, приложения с фрагментами программного кода с комментариями.

Мы предлагаем осуществить реализацию с помощью CMS WordPress. Подробные методические указания и видео расположены в СДО.

Пример реализации № 1

Программная реализация компьютерного тренажера.

Ход работы:

Наш тренажер разбит на 2 части: веб-приложение, которое по запросам выдает страницы с тестами пользователям, и генератор вопросов, который используют разработчики/администраторы для быстрого и легкого составления вопросов к тестам.

Генератор вопросов

Для создания вопросов к тесту было разработано приложение основанное на Windows Forms. При запуске приложения пользователя встретит стартовая форма, с помощью которой пользователь сможет перейти к формам для создания вопросов для легкого и среднего тестов.

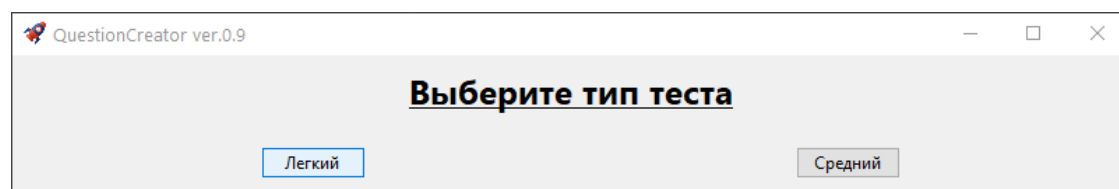


Рисунок 1 – Стартовая форма генератора вопросов

Рассмотрим форму для создания легких вопросов.

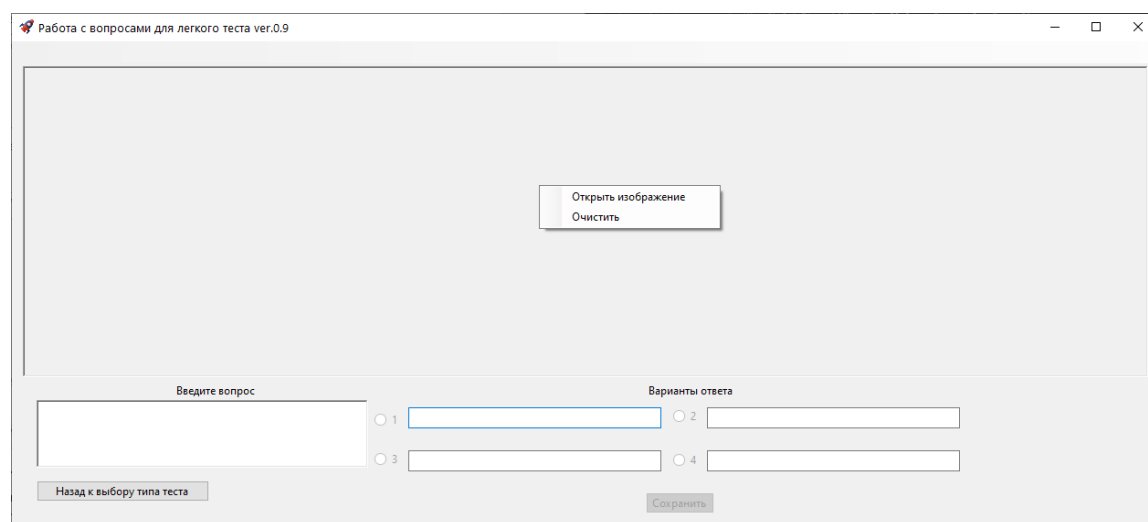


Рисунок 2 – Форма для создания легких вопросов

При нажатии правой кнопкой мыши на окно изображения откроется контекстное меню, с помощью которого пользователь может добавить или убрать изображение.

Чтобы сохранить вопрос пользователю необходимо добавить изображение, ввести вопрос, варианты ответа и выбрать какой вариант является корректным.

Работа с вопросами для легкого теста ver.0.9

Главное здание ГУАП

Введите вопрос

Какому университету принадлежит это здание?

Варианты ответа

☒ 1 Санкт-Петербургский государственный универси

☐ 2 Университет ИТМО

☐ 3 Санкт-Петербургский политехнический универс

☐ 4 Московский государственный университет

Назад к выбору типа теста

Сохранить

Рисунок 3 – Пример заполненной формы для создания легких вопросов

Рассмотрим форму для создания средних вопросов.

Работа с вопросами для среднего теста ver.0.9

Открыть изображение

Очистить

Введите вопрос

Введите возможные ответы

Назад к выбору типа теста

Сохранить


Рисунок 4 – Форма для создания средних вопросов

Средняя форма содержит аналогичные легкой форме элементы. Форма спроектирована чтобы соответствовать модели вопроса.

Для сохранения вопроса, пользователю необходимо добавить изображение, ввести текст и через точку с запятой ввести возможные ответы. После заполнения соответствующих полей, кнопка «Сохранить» будет разблокирована.

Работа с вопросами для среднего теста ver.0.9

guap-lensoveta



Введите вопрос

По какому адресу находится это здание?

Введите возможные ответы

Ленсовета; улица Ленсовета; Ленсовета 14; улица Ленсовета дом 14; в сердечке

Количество возможных ответов: 5

Назад к выбору типа теста Сохранить

Рисунок 5 – Пример заполненной формы для создания средних вопросов

Каждая форма генерирует JSON файл (рисунок 6), который соответствует модели вопроса.

Программный код форм:

1) Стартовая форма(startForm)

```
using System;
using System.Windows.Forms;

namespace QuestionCreator.Forms
{
    public partial class startForm : Form
    {
        public startForm()
        {
            InitializeComponent();
            Text = $"{Application.ProductName}
ver.{Application.ProductVersion}"; // Изменение заголовка формы на основе названия
приложения и его версии
        }

        private void button1_Click(object sender, EventArgs e) //
Обработчик нажатия кнопки для отображения формы для создания легких вопросов
        {
            Easy easy = new Easy();
            easy.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e) //
Обработчик нажатия кнопки для отображения формы для создания средних вопросов
        {
            Medium medium = new Medium();
        }
    }
}
```

```

        medium.Show();
        this.Hide();
    }
}
}

```

2) Форма для создания легких вопросов (easy)

```

using System;
using System.Drawing;
using System.IO;
using System.Windows.Forms;
using Newtonsoft.Json;
using WebApplication.Models.Easy;

namespace QuestionCreator
{
    public partial class Easy : Form
    {
        public Easy()
        {
            InitializeComponent();
            FormClosed += Form_Closed;
            Text = $"Работа с вопросами для легкого теста
ver.{Application.ProductVersion}"; // Изменение заголовка формы на основе версии
приложения

            ToolStripMenuItem openImage = new ToolStripMenuItem("Открыть
изображение");
            ToolStripMenuItem cleanImage = new ToolStripMenuItem("Очистить");

            openImage.Click += openImage_Click;
            cleanImage.Click += cleanImage_Click;

            contextMenuStrip1.Items.AddRange(new[] { openImage, cleanImage }); //
Добавление кнопок в контекстное меню
            pictureBox1.ContextMenuStrip = contextMenuStrip1; // Привязка
контекстного меню к контейнеру с изображением
        }

        Image image;
        bool opened = false;
        string answer;
        string image_location;
        string pictureName;

        public void openImage_Click(object sender, EventArgs e) // Обработчик
кнопки для добавления изображения
        {
            DialogResult dialogResult = openFileDialog1.ShowDialog();
            if (dialogResult == DialogResult.OK)
            {
                image = Image.FromFile(openFileDialog1.FileName);
                pictureBox1.Image = image;
                image_location = openFileDialog1.FileName;

                pictureName =
Path.GetFileNameWithoutExtension(openFileDialog1.FileName);

                label3.Text = pictureName;
                label3.Visible = true;
                opened = true;
            }
        } и т.д.
    }
}

```

Пример № 2

Ход выполнения работы:

Разработка интерфейса:

1. Реализация навигации

1.1. Навигационная панель выполнена в виде выезжающего меню

Реализация меню:

```
import React from 'react';
import classes from './Drawer.module.scss';
import Backdrop from '../UI/Backdrop/Backdrop';
import {NavLink} from 'react-router-dom';

class Drawer extends React.Component {
  renderLinks(links) {
    return links.map((link, index) => {
      return (
        <li
          key={index}
        >
          <NavLink
            to={link.to}
            exact={link.exact}
            activeClassName={classes.active}
            onClick={this.props.onClose}
          >{
            link.label}
          </NavLink>
        </li>
      )
    })
  }
  render () {
    const cls = [classes.Drawer]
    if (!this.props.isOpen) {
      cls.push(classes.close)
    }
  }
}
```

```

    }

    const links = [
      {to: '/', label: 'Главная', exact: true},
    ]

    if (this.props.isAuthenticated) {
      links.push({to: '/quiz-creator', label: 'Создать тест', exact: false})
      links.push({to: '/logout', label: 'Выйти', exact: false})
    } else {
      links.push({to: '/auth', label: 'Авторизация', exact: false})
    }
    return (
      <React.Fragment>
        <nav className={cls.join(' ')}>
          <ul>
            {this.renderLinks(links)}
          </ul>
        </nav>
        {this.props.isOpen ? <Backdrop onClick={this.props.onClose}/> : null }
      </React.Fragment>
    )
  }
}

export default Drawer

```

1.2.Механизм, осуществляющий выдвижение меню:

```

// import React from 'react;'
import classes from './MenuToggle.module.scss';

const MenuToggle = props => {
  const cls = [

```

export default MenuToggle

2. Страница аутентификации:

```
import React from 'react';

import classes from './Auth.module.scss'

import Button from '../components/UI/Button/Button'

import Input from '../components/UI/Input/Input'

import { connect } from 'react-redux';

import { auth } from '../store/actions/auth'


function validateEmail(email) {

    const re = /^(^<()[]\|\\.,;:~s@"]+(\.|^<()[]\|\\.,;:~s@"]+)*)(("[.]+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\]|((\[a-zA-Z\-[0-9]+\]|\.)+[a-zA-Z]{2,}))$;/

    return re.test(String(email).toLowerCase());
}
```



```
}

class Auth extends React.Component {
  state = {
    isFormValid: false,
    formControls: {
      email: {
        value: "",
        type: 'email',
        label: 'Email',
        errorMessage: 'Введите корректный email',
        valid: false,
        touched: false,
        validation: {
          required: true,
          email: true
        }
      },
      pass: {
        value: "",
        type: 'password',
        label: 'Пароль',
        errorMessage: 'Введите корректный пароль',
        valid: false,
        touched: false,
        validation: {
          required: true,
          minLength: 6
        }
      }
    }
  }

  loginHandler = async () => {
    this.props.auth(
      this.state.formControls.email.value,
```

```
        this.state.formControls.pass.value,
        true
    )

}

registerHandler = () => {

    this.props.auth(
        this.state.formControls.email.value,
        this.state.formControls.pass.value,
        false
    )

}

submitHandler (event) {
    event.preventDefault()
}

validateControl(value, validation) {
    if(!validation) {
        return true
    }

    let isValid = true

    if (validation.required) {
        isValid = value.trim() !== "" && isValid
    }

    if (validation.email) {
        isValid = validateEmail(value) && isValid
    }
}
```

```

    if (validation.minLength) {
      isValid = value.length >= validation.minLength && isValid
    }

    return isValid
  }

  onChangeHandler = (event, controlName) => {

    const formControls = {...this.state.formControls}
    const control = {...formControls[controlName]}

    control.value = event.target.value
    control.touched = true
    control.valid = this.validateControl(control.value, control.validation)
    formControls[controlName] = control

    let isValid = true
    Object.keys(formControls).forEach(name => {
      isValid = formControls[name].valid && isValid
    })

    this.setState({formControls, isValid})

  }

  renderInputs() {
    return Object.keys(this.state.formControls).map((controlName, index) => {
      const control = this.state.formControls[controlName]
      return (
        <Input
          key = {controlName + index}
          type = {control.type}
          value = {control.value} и т.д.

```

Пример № 3

Разрабатываемый комплекс приложений: Веб-приложения для сканирования людей и создания их 3d аватаров для дальнейшего снятия мерок.

Предметная область: софт для улучшения жизни человека.

Цель работы: разработать код программы

Порядок выполнения работы:

Скриншот доски Trello с текущим уровнем работы над проектом.

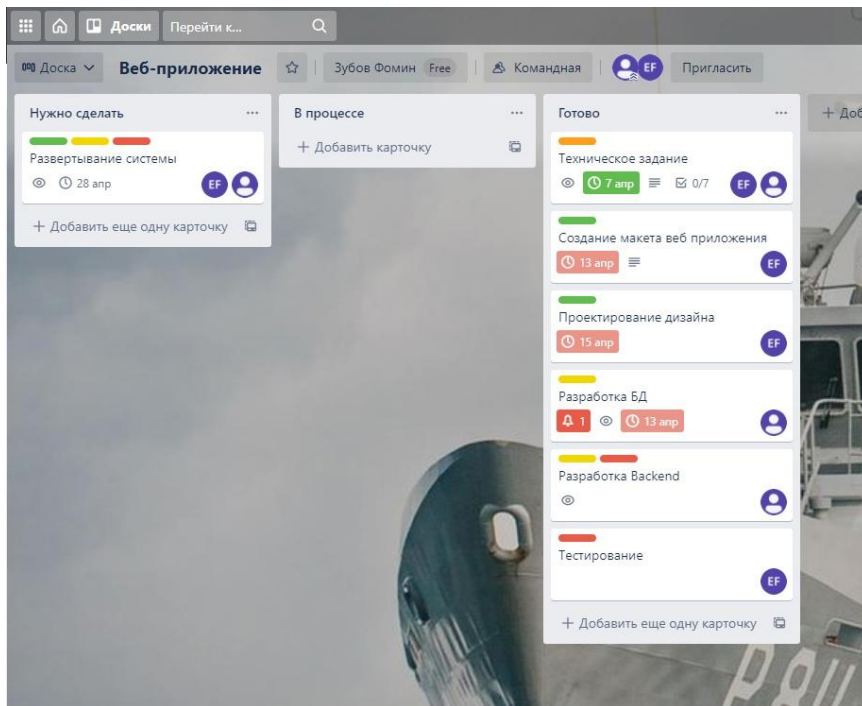


Рисунок 1 – Доска Trello

Далее, предоставляю исходники проекта на сайте gitlab для более удобного прочтения и ознакомления URL: <https://gitlab.com/sewingbackend>

Фрагмент кода:

```
package main

import (
    "context"
    "net/http"
    "time"

    "gitlab.com/sewingbackend/restapi_service/config"
    "gitlab.com/sewingbackend/restapi_service/controller/callbacks"

    "gitlab.com/sewingbackend/restapi_service/auth"
    "gitlab.com/sewingbackend/restapi_service/service"
    "gitlab.com/sewingbackend/restapi_service/store"
    "github.com/pkg/errors"
```

```

    echoLog "github.com/labstack/gommon/log"

    "gitlab.com/sewingbackend/restapi_service/controller"
    libError "gitlab.com/sewingbackend/restapi_service/lib/error"
    "gitlab.com/sewingbackend/restapi_service/lib/validator"
    "gitlab.com/sewingbackend/restapi_service/logger"

    "log"

    "github.com/labstack/echo/v4"
    "github.com/labstack/echo/v4/middleware"
)

func main() {
    if err := run(); err != nil {
        log.Fatal(err)
    }
}

func run() error {
    ctx := context.Background()

    // config
    cfg := config.Get()
    //cfg.HTTPAddr = ":8080"
    //cfg.AmqpURL = "amqp://test:test@localhost:5672"
    //cfg.PgURL =
    "postgres://sartor_user:magical_password@localhost/sartor_database?sslmode=disable"
    //cfg.FilePath = "../.../Storage"
    // - HTTP_ADDR=:8080
    // - FILE_PATH=../.../files
    // -
    PG_URL=postgres://sartor_user:magical_password@database/sartor_database?sslmode=disable
    // - LOG_LEVEL=debug
    // - AMQP_URL=amqp://test:test@rabbitmq:5672

    // logger
    l := logger.Get()

    // Init repository store (with mysql/postgresql inside)
    store, err := store.New(ctx)
    if err != nil {
        return errors.Wrap(err, "store.New failed")
    }

    // Init service manager
    serviceManager, err := service.NewManager(ctx, store)
    if err != nil {
        return errors.Wrap(err, "manager.New failed")
    }

    err = serviceManager.MessagingClinet.ConnectToBroker(cfg.AmqpURL)
    if err != nil {
        return errors.Wrap(err,
            "rabbitmq.MessagingClinet.ConnectToBroker failed")
    }

    //serviceManager.MessagingClinet.PublishOnQueue(,)
    // Init controllers
    vendorController := controller.NewVendors(ctx, serviceManager, l)
    customerController := controller.NewCustomers(ctx, serviceManager, l)

```

```
        measurementsController := controller.NewMeasuremenst(ctx,  
serviceManager, 1)  
  
        //Callbacks  
        servicesMessages := callbacks.NewServicesMessages(ctx, serviceManager,  
1)
```

И Т.Д.