ДОДАТОК А
Текст програми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

"ЗАТВЕРДЖУЮ"
керівник атестаційної роботи
проф. Іванов В.Г.

ПОДАННЯ ІНФОРМАЦІЇ В WEB-СИСТЕМАХ З УРАХУВАННЯ СПРИЙНЯТТЯ І ОСОБИСТИХ ЯКОСТЕЙ УЧНЯ

ПАКЕТ ПРОГРАМ ВИЗНАЧЕННЯ СТРУКТУРИ ОБ'ЄКТА ПРОЕКТУВАННЯ

Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
ГЮИК. 502610.029 -01 12 01-ЛЗ

УЗГОДЖЕНО:

РОЗРОБИЛА:
ст.гр. СПРм-18-1
Канєвська А. Г.

2019

ЗАТВЕРДЖЕНО
ГЮИК. 502610.029 -01 12 01-ЛЗ

ПОДАННЯ ІНФОРМАЦІЇ В WEB-СИСТЕМАХ З УРАХУВАННЯ СПРИЙНЯТТЯ І ОСОБИСТИХ ЯКОСТЕЙ УЧНЯ

ПАКЕТ ПРОГРАМ ВИЗНАЧЕННЯ СТРУКТУРИ ОБ’ЄКТА ПРОЕКТУВАННЯ

Текст програми
ГЮИК. 502610.029 -01 12 01
Аркушів 12

2019

Entry Point

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import
org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
import org.springframework.context.annotation.PropertySource;

@SpringBootApplication
@PropertySource(value = {"classpath:/properties/payment/payment.properties",
        "classpath:/properties/email/email-
${spring.profiles.active}.properties"})
public class StudyProjectApplication extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder
application) {
        return application.sources(StudyProjectApplication.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(StudyProjectApplication.class, args);
    }
}
```

YAML config

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.context.properties.ConfigurationProperties;
import
org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.stereotype.Component;
import java.util.HashMap;

@Component
@EnableConfigurationProperties
@ConfigurationProperties(prefix = "restdata")
public class YAMLConfig {
    private String token;
    private HashMap<String,String> urls;

    public HashMap<String, String> getUrls() {
        return urls;
    }

    public void setUrls(HashMap<String, String> urls) {
        this.urls = urls;
    }

    public String getToken() {
        return token;
    }

    public void setToken(String token) {
        this.token = token;
    }
}
```

```
@Configuration
@PropertySource(value={"classpath:app.properties"}, ignoreResourceNotFound =
true)
public class PropertiesConfig {

    @Bean
    public static PropertySourcesPlaceholderConfigurer properties() {

        PropertySourcesPlaceholderConfigurer
propertySourcesPlaceholderConfigurer = new
PropertySourcesPlaceholderConfigurer();

        YamlPropertiesFactoryBean yaml = new YamlPropertiesFactoryBean();

        yaml.setResources(new ClassPathResource("/appconst.yaml"));

        propertySourcesPlaceholderConfigurer.setProperties(yaml.getObject());

        return propertySourcesPlaceholderConfigurer;
    }
}
```

JSONResponseHandler

```
public class JSONResponseHandler implements ResponseHandler<JSONObject> {
    private static final Logger LOGGER=
LogManager.getLogger(JSONResponseHandler.class);
    @Override
    public JSONObject handleResponse(final HttpResponse response) {
        int status = response.getStatusLine().getStatusCode();
        JSONObject returnData = new JSONObject();
        JSONParser parser = new JSONParser();
        if (status == 401) {
            LOGGER.info("token refreshing");
            returnData.put("status_code", "401");
        } else if (status >= 200 && status < 300) {
            HttpEntity entity = response.getEntity();
            try {
                if (null == entity) {
                    returnData.put("status_code", "1");
                    returnData.put("error_message", "null Data Found");
                    LOGGER.error("null Data Found");

                } else {
                    returnData = (JSONObject)
parser.parse(EntityUtils.toString(entity));
                    returnData.put("status_code", "0");
                    LOGGER.info("getting data success");
                }
            } catch (ParseException | IOException |
org.json.simple.parser.ParseException e) {
                returnData.put("status_code", "1");
                returnData.put("error_message", e.getMessage());
```

```
                LOGGER.error("parsing error");
                LOGGER.info(e.getMessage());
            }

        } else {
            LOGGER.error("Unexpected response status: " + status);
            returnData.put("status_code", "1");
            returnData.put("error_message", "Unexpected response status: " +
status);

        }
        return returnData;
    }
```

HttpClientHelperImpl

```
@Component
public class HttpClientHelperImpl implements HttpClientHelper {

    private ResponseHandler<JSONObject> responseHandler =
(ResponseHandler<JSONObject>) new JSONResponseHandler();
    private Map<String, String> headers = new HashMap<>();
    private Map<String, String> params = new HashMap<>();
    private String paramsList ;
    private ArrayList<NameValuePair> postParameters = new ArrayList<>();

    @Override
    public void addToHeaders(String key, String value) {
        this.headers.put(key, value);
    }

    @Override
    public void addToBodyParams(String key, String value) {
        this.postParameters.add(new BasicNameValuePair(key, value));
    }

    @Override
    public void addToBodyJSON(String key, String value) {
        this.params.put(key, value);
    }

    @Override
    public void addToBodyJSON(String value) {
        this.paramsList=value;
    }

    @Override
    public JSONObject sendGet(String url) throws TokenException {
        try (CloseableHttpClient httpclient = HttpClients.createDefault()) {
            HttpGet httpget = new HttpGet(url);
            if (!headers.isEmpty()) {
                for (Map.Entry<String, String> entry : headers.entrySet()) {
                    httpget.setHeader(entry.getKey(), entry.getValue());
                }
            }

            JSONObject responseBody = httpclient.execute(httpget,
responseHandler);
```

```
            String statusCode = (String) responseBody.get("status_code");
            if (statusCode.equalsIgnoreCase("0")) {
                return responseBody;
            } else if (statusCode.equalsIgnoreCase("401")) {
                throw new TokenException();
            } else {
                //TODO....
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }

    @Override
    public JSONObject sendPost(String url) throws TokenException {
        try (CloseableHttpClient httpclient = HttpClients.createDefault()) {
            HttpPost httpost = new HttpPost(url);

            if (!params.isEmpty()) {
                StringEntity stringEntity = new StringEntity(new
Gson().toJson(params));
                httpost.setEntity(stringEntity);
            }
            if (paramsList!=null) {
                StringEntity stringEntity = new StringEntity(paramsList);
                httpost.setEntity(stringEntity);
            }

            if (!headers.isEmpty()) {
                for (Map.Entry<String, String> entry : headers.entrySet()) {
                    httpost.setHeader(entry.getKey(), entry.getValue());
                }
            }

            if (!postParameters.isEmpty()) {
                httpost.setEntity(new UrlEncodedFormEntity(postParameters, "UTF-
8"));
            }
            JSONObject responseBody = httpclient.execute(httpost,
responseHandler);

            String statusCode = (String) responseBody.get("status_code");

            if (statusCode.equalsIgnoreCase("0")) {
                httpclient.close();
                return responseBody;
            }else if (statusCode.equalsIgnoreCase("401")) {
                throw new TokenException();
            } else {
                //TODO....
            }
            System.out.println(responseBody.get("error_message"));
        } catch (IOException e) {
            e.printStackTrace();
        }

        return null;
   }}
```

InformationServiceImpl

```
@Import(PropertiesConfig.class)
@Service
public class InformationServiceImpl implements InformationService {
    private static final Logger LOGGER =
LogManager.getLogger(InformationServiceImpl.class);
    @Autowired
    HttpClientHelper httpClientHelper;
    private GsonBuilder gb = new GsonBuilder().registerTypeAdapter(Date.class,
new DateDeserializerForContracts());
    private Gson gson = gb.serializeNulls().create();


    private @Value("${restdata.urls.chart}")
    String chartUrl;
    private @Value("${restdata.urls.userdata}")
    String userDataUrl;
    private @Value("${restdata.urls.services}")
    String servicesUrl;

    @Override
    public List<Chart> getChartInfo(String clientId, String token) throws
TokenException {

        if (token == null) {
            throw new TokenException();
        }
        httpClientHelper = new HttpClientHelperImpl();
        httpClientHelper.addToHeaders(HttpHeaders.AUTHORIZATION, "Bearer " +
token);
        String url = String.format(chartUrl, clientId);
        JSONObject response = (httpClientHelper.sendGet(url));
        String list = "";
        if (response != null) {
            list = response.get("List").toString();
        } else {
            return new ArrayList<>();
        }
        Type type = new TypeToken<ArrayList<Chart>>() {
        }.getType();

        return gson.fromJson(list, type);

    }

    @Override
    public UserData getUserData(String clientId, String token) throws
TokenException {

        if (token == null) {
            throw new TokenException();
        }

        httpClientHelper = new HttpClientHelperImpl();
        httpClientHelper.addToHeaders(HttpHeaders.AUTHORIZATION, "Bearer " +
token);
        String url = String.format(userDataUrl, clientId);
        JSONObject resp = (httpClientHelper.sendGet(url));
        if (resp != null) {
            JSONObject response =(JSONObject)resp.get("response");
```

```
            JSONObject claster =(JSONObject) response.get("cluster");
            UserData userData = gson.fromJson(response.toString(),
UserData.class);
            userData.setCluster(claster.get("value").toString());
            return userData;
        }
        return new UserData();

    }

    @Override
    public List<Contracts> getContractsData(String clientId, String token)
throws TokenException {

        if (token == null) {
            throw new TokenException();
        }
        httpClientHelper = new HttpClientHelperImpl();
        httpClientHelper.addToHeaders(HttpHeaders.AUTHORIZATION, "Bearer " +
token);
        httpClientHelper.addToHeaders(HttpHeaders.CONTENT_TYPE,
"application/json");

        JSONObject response = httpClientHelper.sendPost(DataUrl);
        if (response != null) {
            response = (JSONObject) response.get("response");
        } else {
            return new ArrayList<>();
        }
        String array = response.get("contratti").toString();
        Type type = new TypeToken<ArrayList<Contracts>>() {
        }.getType();

        return gson.fromJson(array, type);
    }

    @Override
    public List< Service> getServices(String clientId, String token) throws
TokenException {
        if (token == null) {
            throw new TokenException();
        }
        httpClientHelper = new HttpClientHelperImpl();
        httpClientHelper.addToHeaders(HttpHeaders.AUTHORIZATION, "Bearer " +
token);
        String url = String.format(servicesUrl, clientId);
        JSONObject response = (httpClientHelper.sendGet(url));
        if (response == null) {
            return new ArrayList<>();
        }
        response = (JSONObject) response.get("response");
        String array = response.get("servizio").toString();
        Type type = new
TypeToken<ArrayList<com.optimaitalia.model.services.Service>>() {
        }.getType();

        GsonBuilder gb = new GsonBuilder().registerTypeAdapter(Date.class, new
DateDeserializerForContracts());
        Gson gson = gb.create();
        return gson.fromJson(array, type);
```

```
    }
```

app.component.ts

```
import { Component, HostBinding, OnInit } from '@angular/core';
declare var $: any;

import { SettingsService } from './core/settings/settings.service';

@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.scss']
})
export class AppComponent implements OnInit {

    @HostBinding('class.layout-fixed') get isFixed() { return
this.settings.layout.isFixed; }
    @HostBinding('class.aside-collapsed') get isCollapsed() { return
this.settings.layout.isCollapsed; }
    @HostBinding('class.layout-boxed') get isBoxed() { return
this.settings.layout.isBoxed; }
    @HostBinding('class.layout-fs') get useFullLayout() { return
this.settings.layout.useFullLayout; }
    @HostBinding('class.hidden-footer') get hiddenFooter() { return
this.settings.layout.hiddenFooter; }
    @HostBinding('class.layout-h') get horizontal() { return
this.settings.layout.horizontal; }
    @HostBinding('class.aside-float') get isFloat() { return
this.settings.layout.isFloat; }
    @HostBinding('class.offsidebar-open') get offsidebarOpen() { return
this.settings.layout.offsidebarOpen; }
    @HostBinding('class.aside-toggled') get asideToggled() { return
this.settings.layout.asideToggled; }
    @HostBinding('class.aside-collapsed-text') get isCollapsedText() { return
this.settings.layout.isCollapsedText; }

    constructor(public settings: SettingsService) { }

    ngOnInit() {
        $(document).on('click', '[href="#"]', e => e.preventDefault());
    }
}
```

app.modlle.ts

```
export function createTranslateLoader(http: HttpClient) {
  return new TranslateHttpLoader(http, './assets/i18n/', '.json');
}
```

```
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    HttpClientModule,
    BrowserAnimationsModule,
    CoreModule,
    LayoutModule,
    SharedModule.forRoot(),
    RoutesModule,
    MatButtonModule,
    MatCheckboxModule,
    TranslateModule.forRoot({
      loader: {
        provide: TranslateLoader,
        useFactory: (createTranslateLoader),
        deps: [HttpClient]
      }
    })
  ],
  providers: [HttpService,

  ],
  bootstrap: [AppComponent]
})
export class AppModule {
}
```

route.ts

```
export const routes = [

  {
    path: '',
    component: NavigationComponent,
    children: [
      {
        path: '',
        redirectTo: '/login',
        pathMatch: 'full'
      },
      {
        path: 'home',
        loadChildren: './home/home.module#HomeModule'
      },
      {
        path: 'profile',
        loadChildren: './profilePage/profilePage.module#ProfilePageModule'
      },
      {
        path: 'support',
        loadChildren: './answer-questions/answer-
questions.module#AnswerQuestionsModule'
      },
      {
        path: 'services',
        loadChildren: './active-services/active-
services.module#ActiveServicesModule'
      },
```

```
      {path: 'login', loadChildren: './pages/pages.module#PagesModule'}
    ]
  },

  {path: '**', redirectTo: '/home'}

];
```

home.resolver.ts

```
@Injectable()
export class HomeResolver implements Resolve<any> {

  constructor(private service: UserServicesService ) {}

  resolve(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ) {
    return this.service.getUserData();
  }
}
```

login.component.ts

```
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})


export class LoginComponent implements OnInit {

  valForm: FormGroup;
  router: Router;
  public mess;
  private url = 'api/getCredentials';

  constructor(public settings: SettingsService, fb: FormBuilder, private
_router: Router, private httpService: HttpService) {
    this.router = _router;
    this.valForm = fb.group({
      'username': [null, Validators.required],
      'password': [null, Validators.required]


    });

  }


  login(fields) {
    this.httpService.get(this.url, fields).subscribe(
      data => {
        localStorage.setItem('clientId', data['id']);
        this.router.navigate(['/home'], {replaceUrl: true});
        return true;
      },
      error => {
```

```
        console.error('Error');
        this.mess = 'login or password is incorrect';
        return 'error';
      }
    );
  }
  submitForm($ev, value: any) {
    $ev.preventDefault();
    for (const c in this.valForm.controls) {
      this.valForm.controls[c].markAsTouched();
    }
    if (this.valForm.valid) {
      console.log('Valid!');
      console.log(value);
      this.settings.app.username = value.username;
      localStorage.setItem('clientUserName', value.username);
      this.login(value);

    } }  ngOnInit() {  }

                                          }
```

ДОДАТОК Б
Графічний матеріал кваліфікаційної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

"ЗАТВЕРДЖУЮ"
керівник атестаційної роботи
проф. Іванов В.Г.

ПОДАННЯ ІНФОРМАЦІЇ В WEB-СИСТЕМАХ З УРАХУВАННЯ СПРИЙНЯТТЯ І ОСОБИСТИХ ЯКОСТЕЙ УЧНЯ

ПАКЕТ ПРОГРАМ ВИЗНАЧЕННЯ СТРУКТУРИ ОБ'ЄКТА ПРОЕКТУВАННЯ

Графічний матеріал аткстайної роботи

ЛИСТ ЗАТВЕРДЖЕННЯ

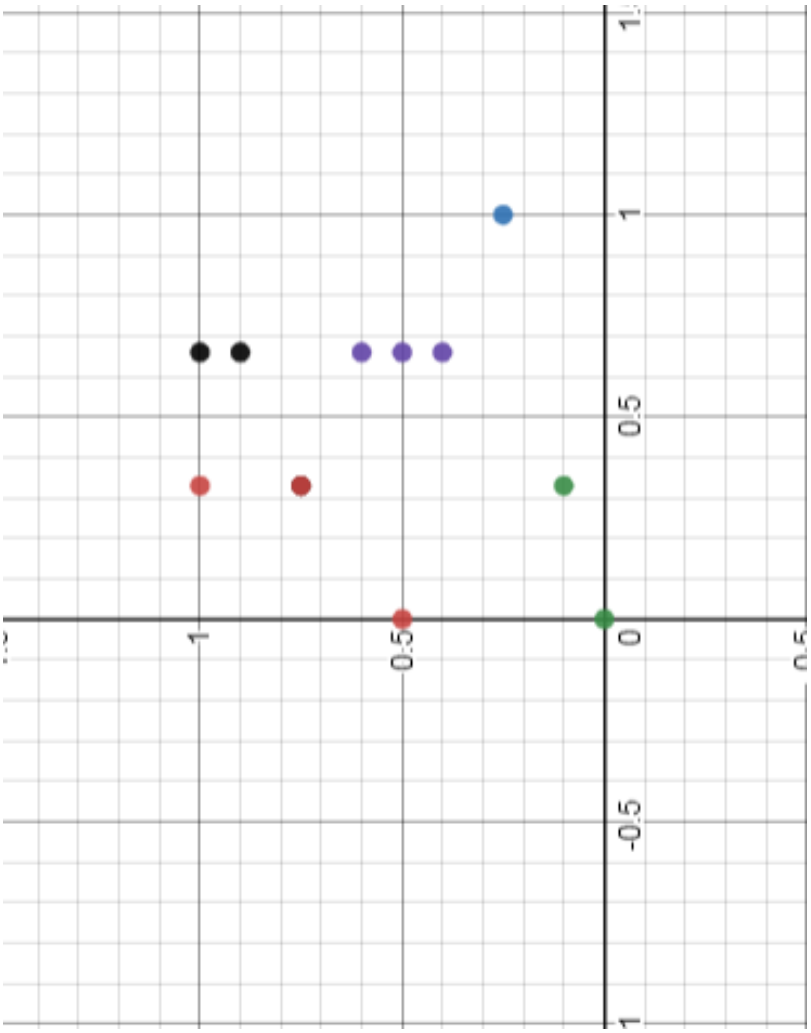
ГЮИК. 502610.029-01 12 01-ЛЗ

УЗГОДЖЕНО:

РОЗРОБИЛА:
ст.гр. СПРм-18-1
Канєвська А. Г.

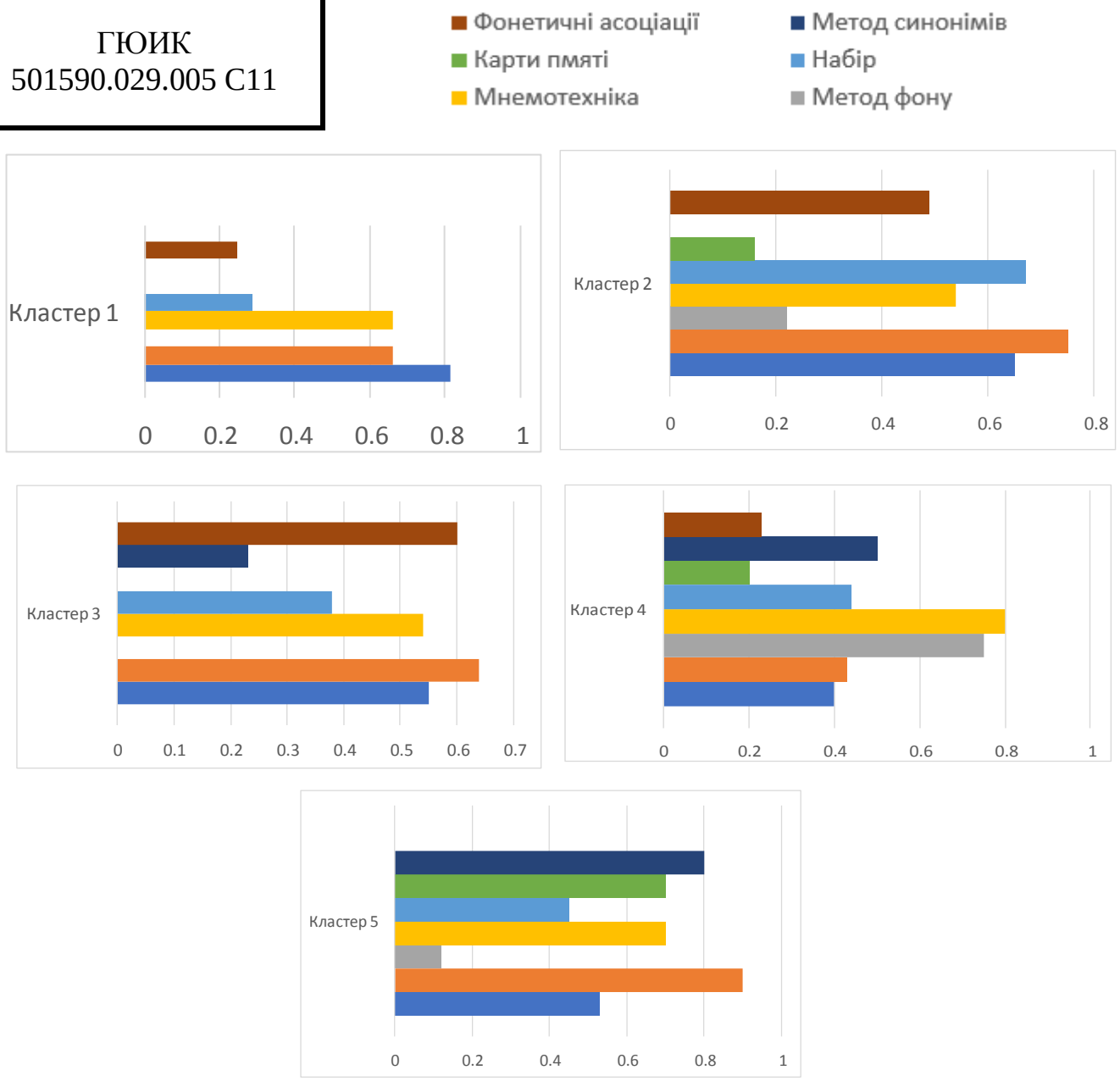
2019

ГЮИК
501590.029.005 С11



					ГЮИК 501590.029 С11	
					Подання інформації в web-системах з урахуванням сприйняття і особистих якостей учня	Арк. / Маса / Масштаб
Ізм..	Арк.	№ докум.	Підпис	Дата		
Розроб.		Канєвська А.Г.				Аркуш 1 / Аркушів 1
Перевір.		Іванов В.Г.				
Т. Контр.					Графічне подання кластерів	ХНУРЕ Кафедра СТ
Реценз.						
Н. Контр.		Іванов В.Г.				
Затверд.		Гребеннік І.В.				

ГЮИК
501590.029.005 С11

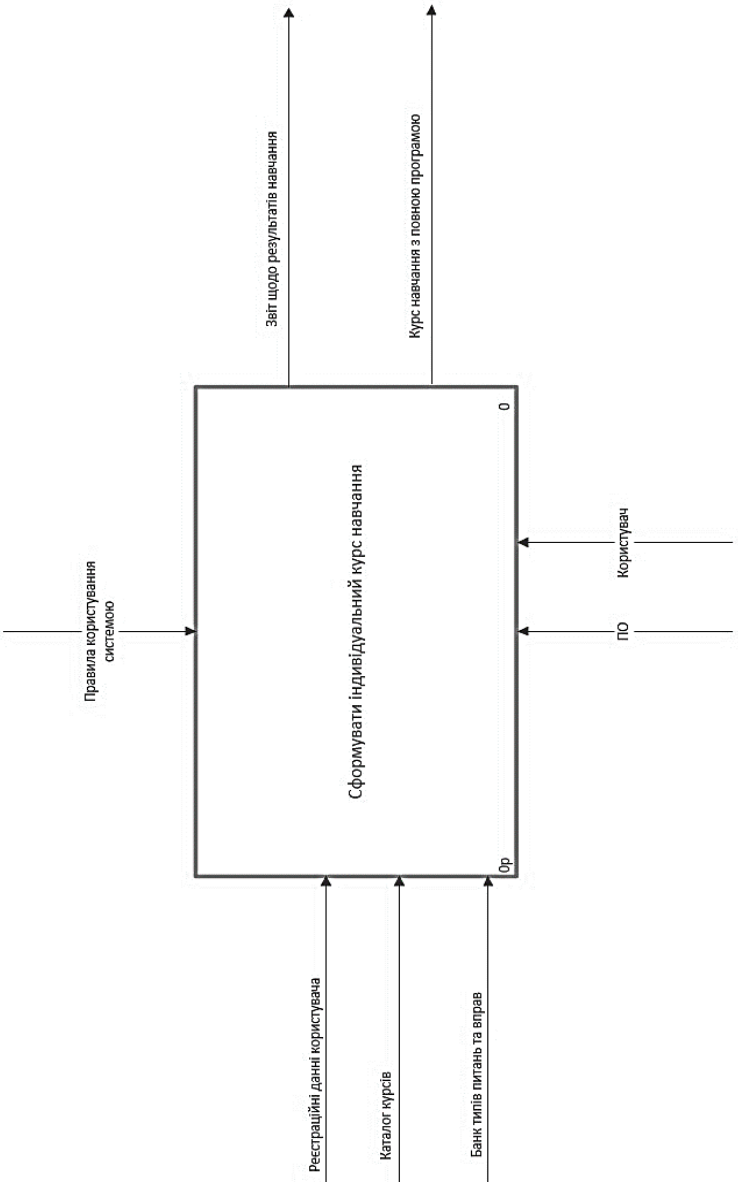


					ГЮИК 501590.029 С11
Ізм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	
Розроб.		Канєвська А.Г.			*Подання інформації в web-системах з урахуванням сприйняття і особистих якостей учня*
Перевір.		Іванов В.Г.			
Т. Контр.					
Реценз.					*Графічне подання резлуьтату експерименту*
Н. Контр.		Іванов В.Г.			
Затверд.		Гребеннік І.В.			

Арк.	*Маса*	*Масштаб*
Аркуш 1	*Аркушів 1*	

ХНУРЕ Кафедра СТ

ГЮИК
501590.029.005 С11

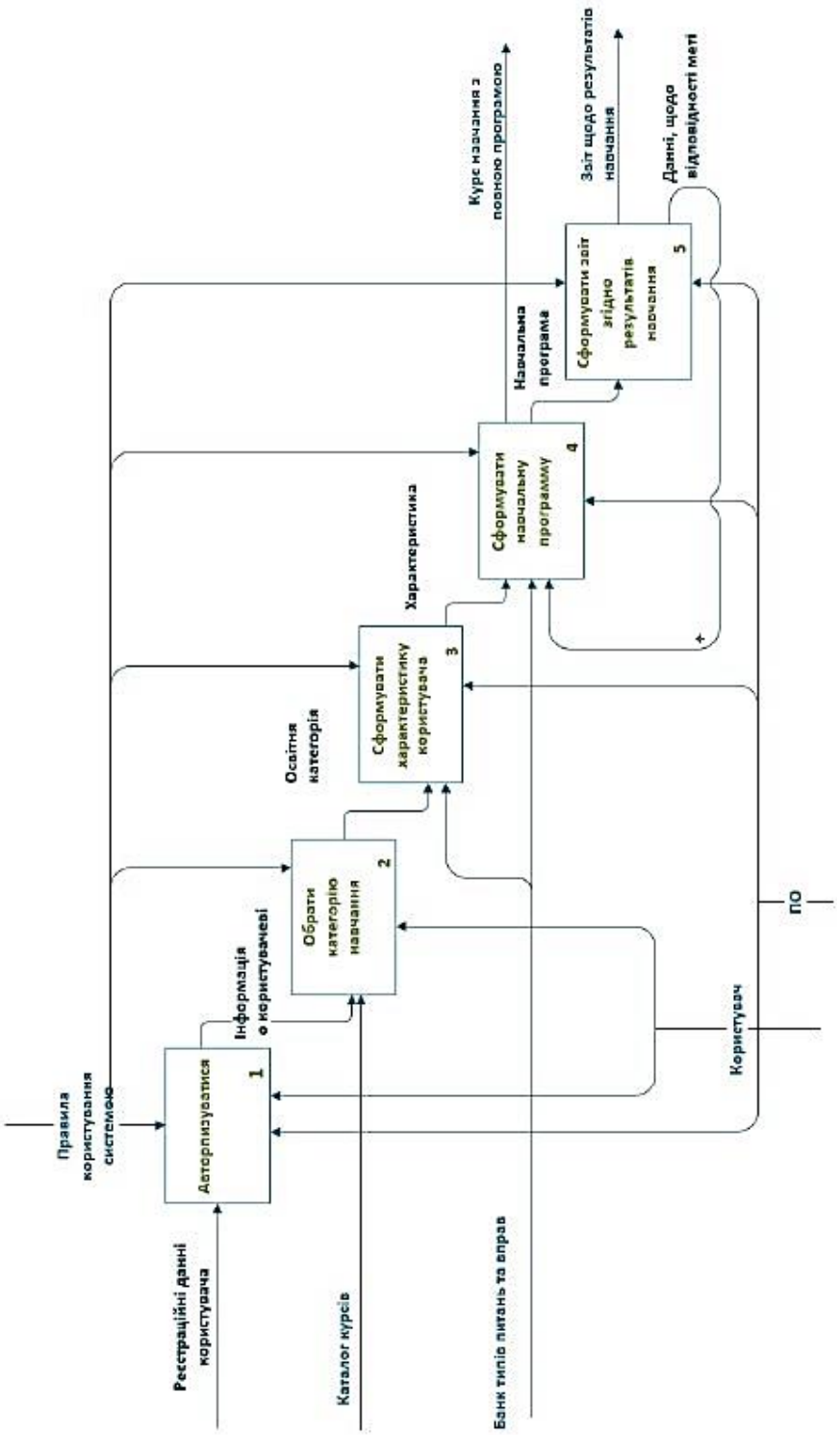


					ГЮИК 501590.029 С11
Ізм..	*Арк.*	*№ докум.*	*Підпис*	*Дата*	
Розроб.		*Канєвська А.Г.*			*Подання інформації в web-системах з урахуванням сприйняття і особистих якостей учня*
Перевір.		*Іванов В.Г.*			
Т. Контр.					
Рецензія.					
Н. Контр.		*Іванов В.Г.*			*Контекстна діаграма формування індивідуального курсу навчання*
Затверд.		*Гребеннік І.В.*			

Арк.	*Маса*	*Масштаб*
Аркуш 1	*Аркушів 3*	

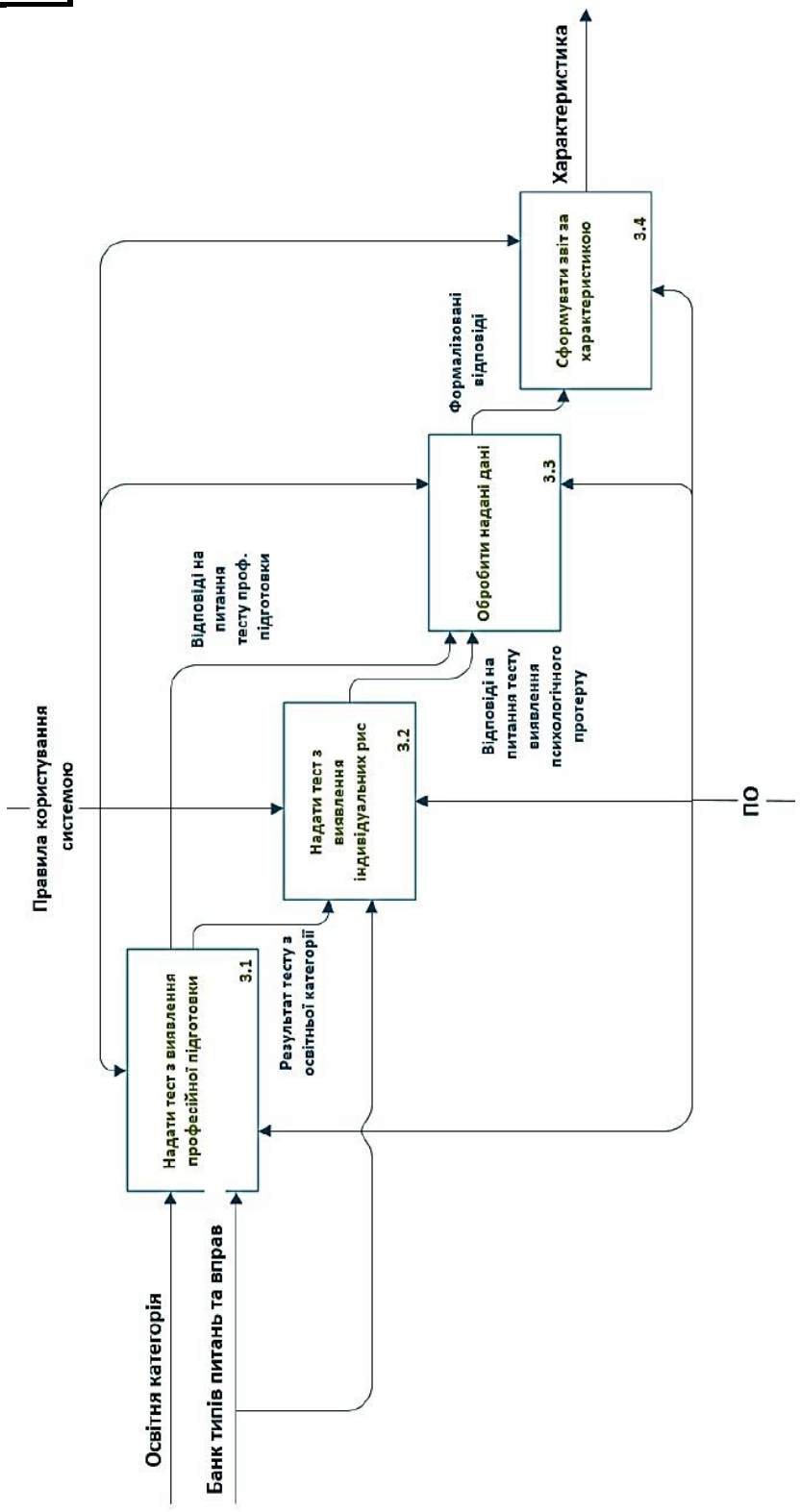
ХНУРЕ Кафедра СТ

ГЮИК
501590.029.005 С11

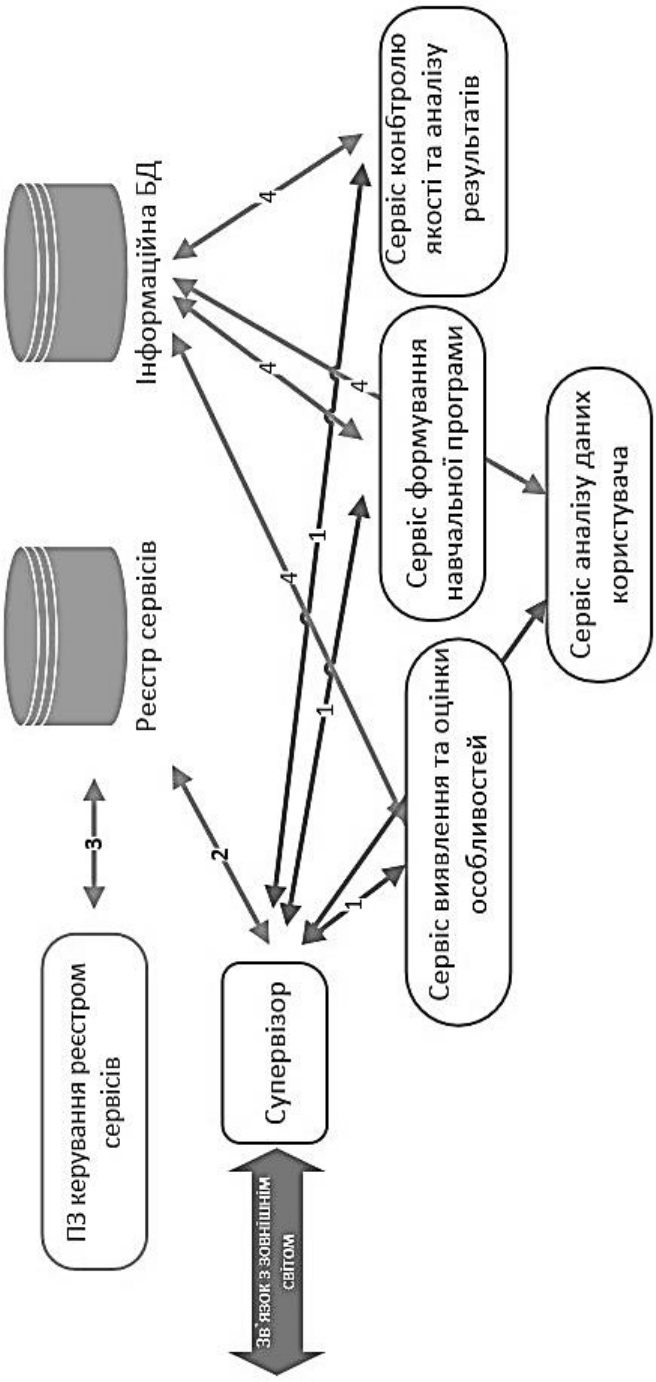


					IDEF0 діаграма декомпозиції блоку формування індивідуальної програми	Аркуш
						2
Ізм..	Арк.	№ докум.	Підпис	Дата		

ГЮИК
501590.029.005 С11

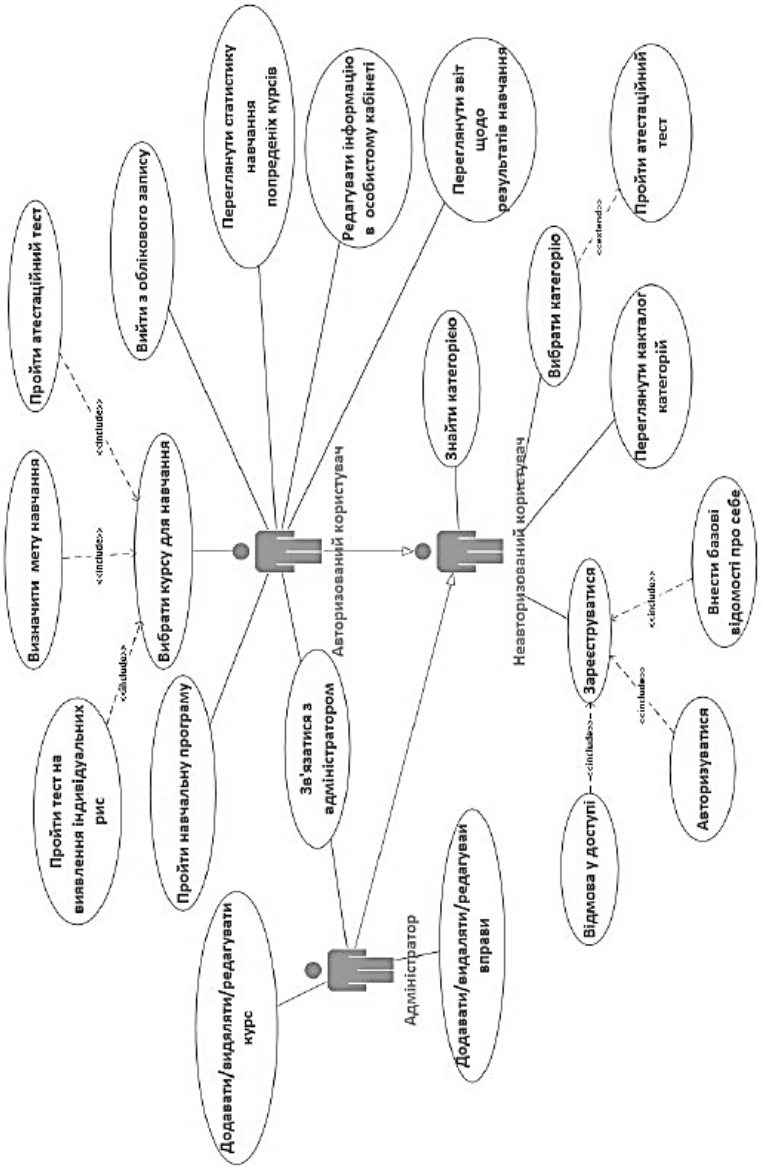


					IDEF0 діаграма декомпозиції блоку формування характеристики користувача	*Аркуш*
						3
Ізм..	*Арк.*	*№ докум.*	*Підпис*	*Дата*		



Розроб.	*Канєвська А. Г.*			*Структурна схема системи формування навчального плану*	
Перевір.	*Іванов В.Г.*				
Н. Контр.	*Іванов В.Г.*				
				СПРм-18-1	*Аркуш 1*
Затверд.	*Гребеннік І.В.*			СТ	*Аркушів 1*

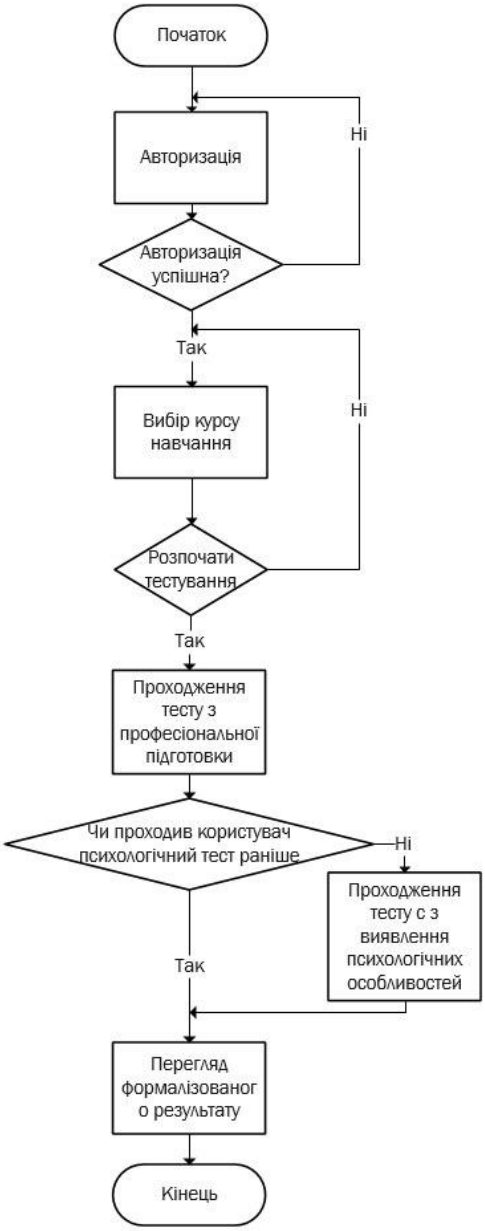
ГЮИК
501590.029.005 С11



					ГЮИК 501590.029 С11
Ізм..	Арк.	№ докум.	Підпис	Дата	Подання інформації в web-системах з урахуванням сприйняття і особистих якостей учня
Розроб.		Канєвська А.Г.			
Перевір.		Іванов В.Г.			
Т. Контр.					
Реценз.					
Н. Контр.		Іванов В.Г.			Діаграма варіантів використання систеи.
Затверд.		Гребеннік І.В.			ХНУРЕ Кафедра СТ

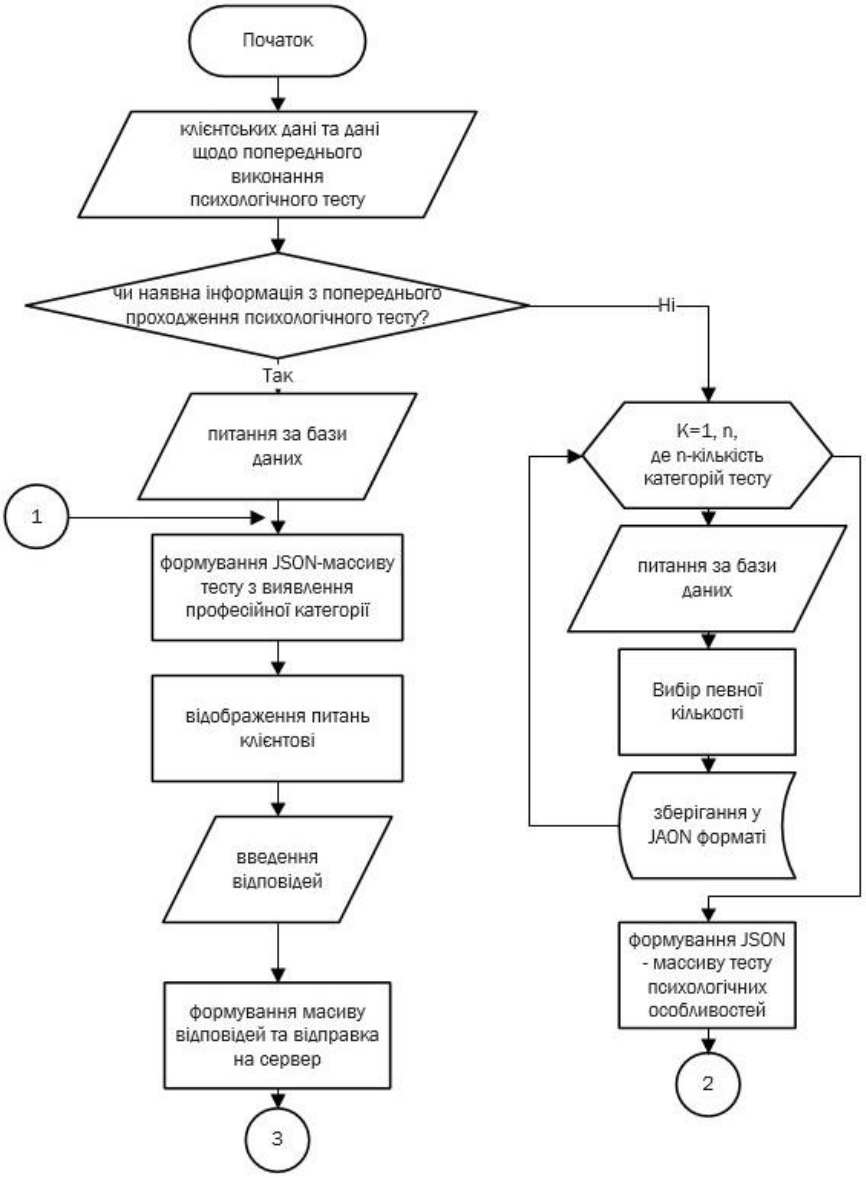
Арк.	Маса	Масштаб
Аркуш 1	Аркушів 3	

ГЮИК
501590.029.005 С11



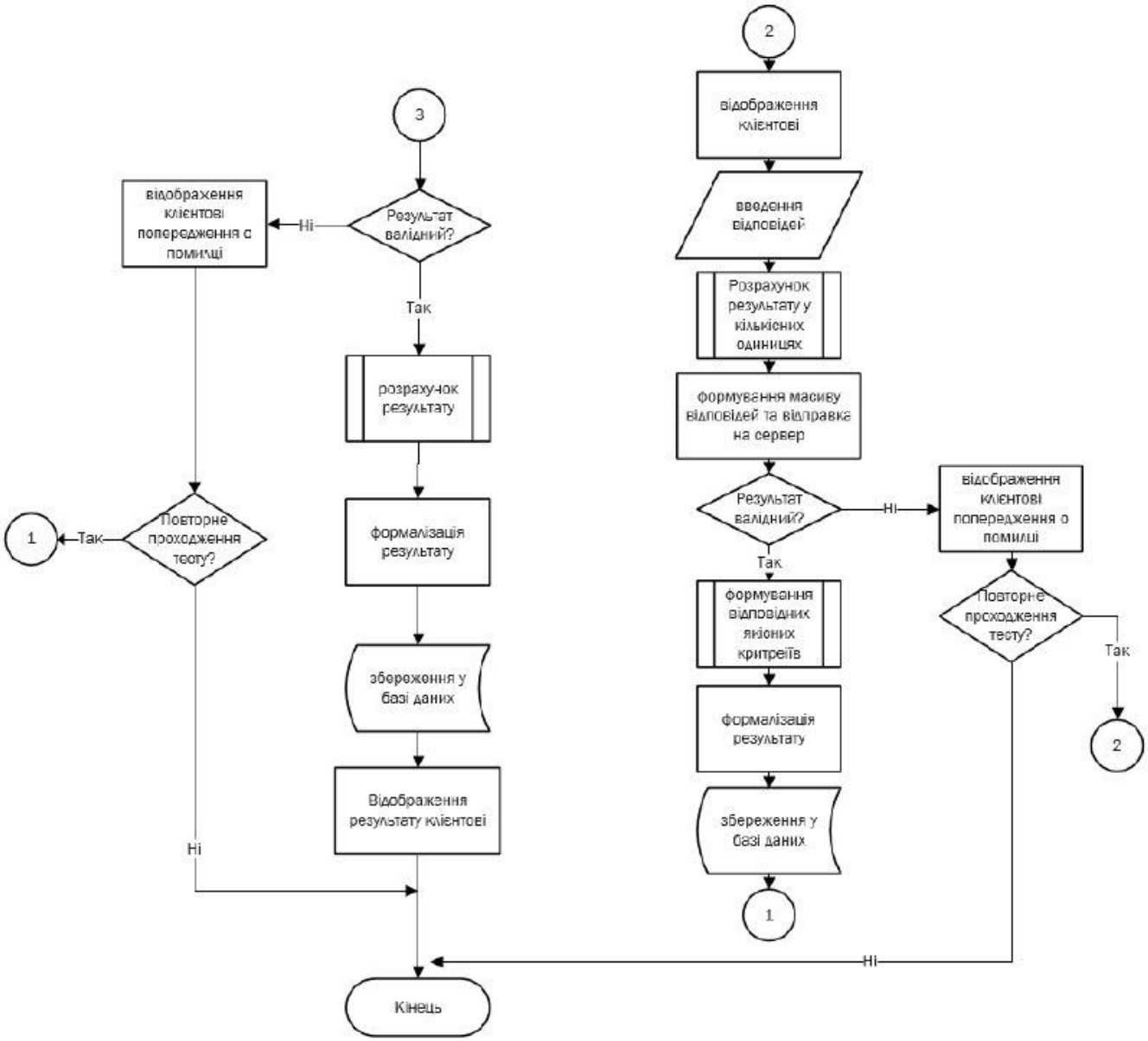
					ГЮИК 501590.029 С11	
Ізм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	*Подання інформації в web-системах з урахуванням сприйняття і особистих якостей учня*	*Арк.* / *Маса* / *Масштаб*
Розроб.		*Канєвська А.Г.*				
Перевір.		*Іванов В.Г.*				*Аркуш 1* / *Аркушів 1*
Т. Контр.						
Реценз.					*Блок-схема алгоритму користування сервісом виявлення характеристик*	*ХНУРЕ Кафедра СТ*
Н. Контр.		*Іванов В.Г.*				
Затверд.		*Гребеннік І.В.*				

ГЮИК
501590.029.005 С11

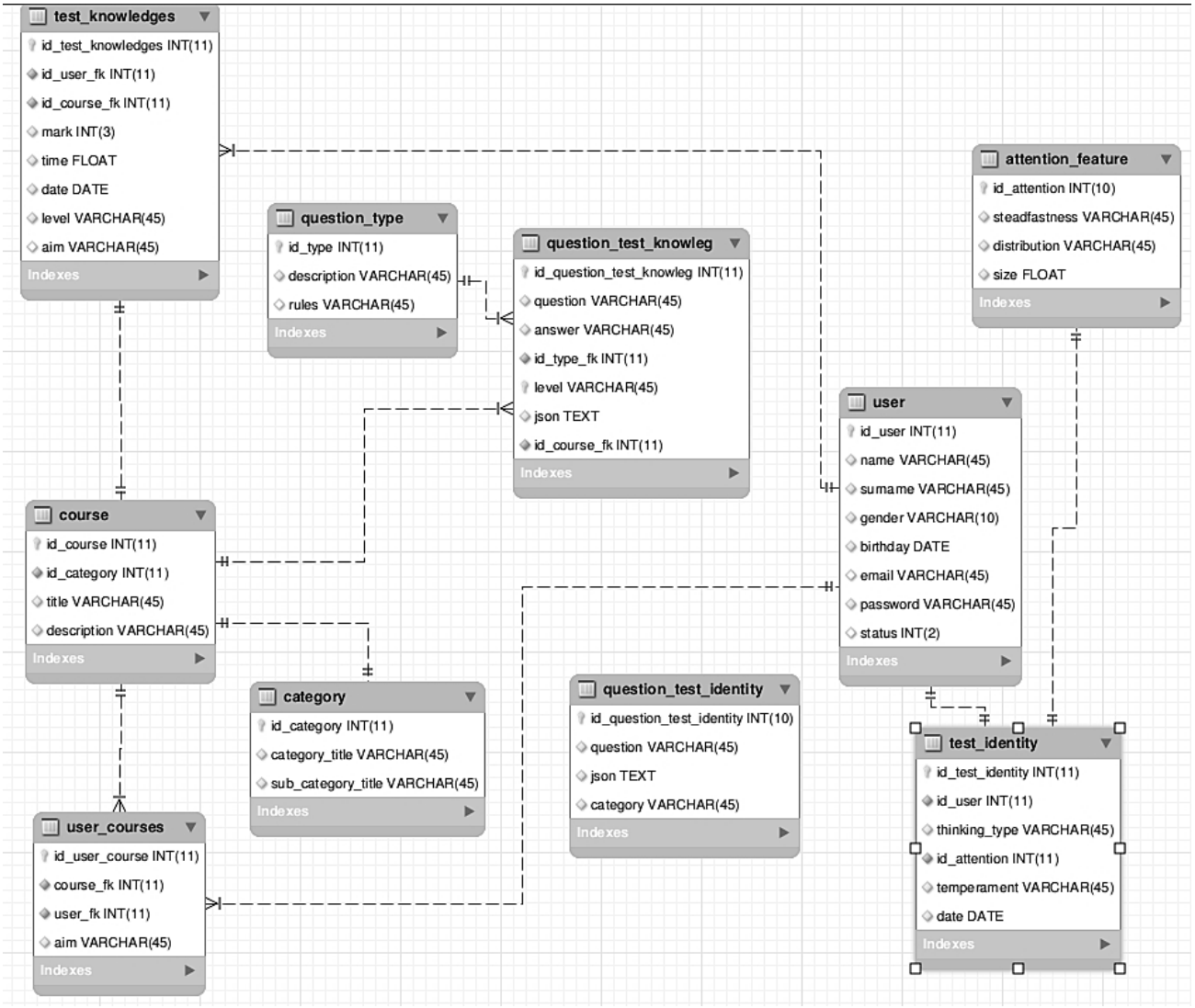


					ГЮИК 501590.029 С11	
Ізм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	*Подання інформації в web-системах з урахуванням сприйняття і особистих якостей учня*	*Арк.* / *Маса* / *Масштаб*
Розроб.		*Канєвська А.Г.*				
Перевір.		*Іванов В.Г.*				*Аркуш 1* / *Аркушів 2*
Т. Контр.						
Реценз.					*Блок-схема алгоритму функціонування сервісу виявлення особливостей користувача*	*ХНУРЕ Кафедра СТ*
Н. Контр.		*Іванов В.Г.*				
Затверд.		*Гребеннік І.В.*				

ГЮИК
501590.029.005 С11



					Продовження алгоритму функціонування сервісу виявлення особливостей користувача	*Аркуш*
						2
Ізм..	*Арк.*	*№ докум.*	*Підпис*	*Дата*		



Розроб.	*Канєвська А. Г.*			*Фізична модель інформаційної бази даних*	
Перевір.	*Іванов В.Г.*				
Н. Контр.	*Іванов В.Г.*				
				СПРм-118-1	*Аркуш 1*
Затверд.	*Гребеннік І.В.*			СТ	*Аркушів 1*