|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное автономное образовательное учреждение высшего образования  «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ» |

Кафедра \_\_\_\_\_\_\_компьютерных технологий и программной инженерии\_\_\_\_\_\_\_\_\_\_\_\_

(наименование)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ОТЧЁТ ПО ПРАКТИКЕ  ЗАЩИЩЁН С ОЦЕНКОЙ  Руководитель |  | | | |
| Ст. преподаватель |  |  |  | М.Д. Поляк |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |
| --- | --- | --- |
| вид практики | производственная | |
| тип практики | научно-исследовательская | |
| на тему индивидуального задания | | Разработка REST API на основе личного кабинета |
| ГУАП | | | |
|  | | | |

|  |  |
| --- | --- |
| выполнен | Беловым Антоном Игоревичем |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.03.04 |  | Программная инженерия |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности | 09.03.04.01 |  | Проектирование программных систем |
|  | код |  | наименование направленности |
|  | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 4932 |  |  |  | А.И. Белов |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2022

Оглавление

[1. Задание 3](#_Toc109073294)

[2. Описание решения 3](#_Toc109073295)

[3. Полученные результаты 8](#_Toc109073296)

[4. Выводы 10](#_Toc109073297)

[5. Список использованной литературы 10](#_Toc109073298)

[6. Приложения 11](#_Toc109073299)

# Задание

**Цель работы:** разработка REST API сервиса для работы с личным кабинетом ГУАП от лица студента.

**Задачи:**

1. Реализовать необходимый для студента функционал
2. Реализовать более простую систему аутентификации
3. Привести все данные к единой системе наименований и стилю CamelCase
4. Привести все получаемые и отправляемые данные к формату JSON
5. Реализовать доступную и удобную документацию
6. Скрыть от пользователя все системные и неиспользуемые данные

# Описание решения

Для выполнения поставленных задач была использована платформа ASP.NET Core и язык C#. В качестве веб-сервера выступает IIS Server. Для тестирования функционала был использован Postman.

1. Были реализованы следующие методы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Путь | Параметры | Возвращаемое значение | Метод | Краткое описание |
| /Auth/Login | Тело:  LoginData data  - данные для авторизации в лк ГУАП | Bool  Успешно ли проведена авторизация | POST | Аутентификация в системе |
| /Auth/Logout | - | Bool  Успешен ли выход из аккаунта | GET | Выход из системы |
| /Material/GetList | int semesterId – необязательно, id необх. семестра  int subjectId – необязательно, id необх. дисциплины | List<Material>  Список материалов | GET | Получить список материалов, по умолчанию за текущий семестр |
| /Material/GetFile | string materialHash – хэш необходимого файла | Byte[]  Файл материала | GET | Скачать приложенный файл |
| /Profile/Get | - | ProfileInfo  Информация профиля | GET | Получить данные о профиле текущего пользователя |
| /Profile/Get  EducationPlan | - | Byte[]  Файл плана обучения | GET | Скачать план обучения текущего пользователя |
| /Subject/GetList | int semesterId – необязательно, id необх. семестра  Int controlTypeId – необязательно, id необх. типа контроля | List<SubjectListItem>  Список дисциплин | GET | Получить список дисциплин, по умолчанию за текущий семестр |
| /Subject/Get | int id – id необходимой дисциплины | Subject  Информация о дисциплине | GET | Получить информацию о дисциплине |
| /Subject/GetTasks | int id – id необходимой дисциплины | List<SubjectTask>  Список заданий к дисциплине | GET | Получить список заданий к дисциплине |
| /Subject/GetAnnotation | int id – id необходимой дисциплины | Byte[]  Файл аннотации к дисциплине | GET | Скачать файл аннотации к дисциплине |
| /Subject/GetEducationPlan | int id – id необходимой дисциплины | Byte[]  Файл плана обучения к дисциплине | GET | Скачать план обучения к дисциплине |
| /Subject/GetWorkProgramm | int id – id необходимой дисциплины | Byte[]  Файл рабочей программы дисциплины | GET | Скачать рабочую программу дисциплины |
| /Subject/GetMaterials | int id – id необходимой дисциплины | List<SubjectMaterial>  Список материалов к дисциплине | GET | Получить список материалов к дисциплине |
| /Task/GetTasks | int semesterId – необязательно, id необх. семестра  Int controlTypeId – необязательно, id необх. типа контроля | List<TaskListItem>  Список заданий | GET | Получить список заданий, по умолчанию за текущий семестр |
| /Task/GetTask | int semesterId – необязательно, id необх. семестра  Int subjectId – необязательно, id необх. дисциплины  int typeId – необязательно, id необх. типа задания  Int statusId – необязательно, id необх. типа статуса | Task  Информация о задании | GET | Получить информацию о задании |
| /Task/GetMaterial | int id – id необходимого задания | Byte[]  Файл доп. материалов к заданию | GET | Скачать доп. материалы к заданию |
| /Task/GetReports | int id – id необходимого задания | List<Report>  Список отчетов к заданию |  | Получить список отчетов к заданию |
| /Task/GetReport | string reportHash – хэш необходимого отчета | Byte[]  Файл отчета к заданию | GET | Скачать файл отчета |
| /Task/AddReport | int taskId – id необходимого задания  тело:  NewReport report – данные добавляемого отчета | Report  Информация о добавленном отчете | POST | Добавить новый отчет |
| /Task/DeleteReport | int reportId – id удаляемого отчета | Bool  Успешно ли удаление отчета | DELETE | Удалить отчет |
| /Teachers/Get | int id – id преподавателя | Teacher  Информация о преподавателе | GET | Получить информацию о преподавателе |

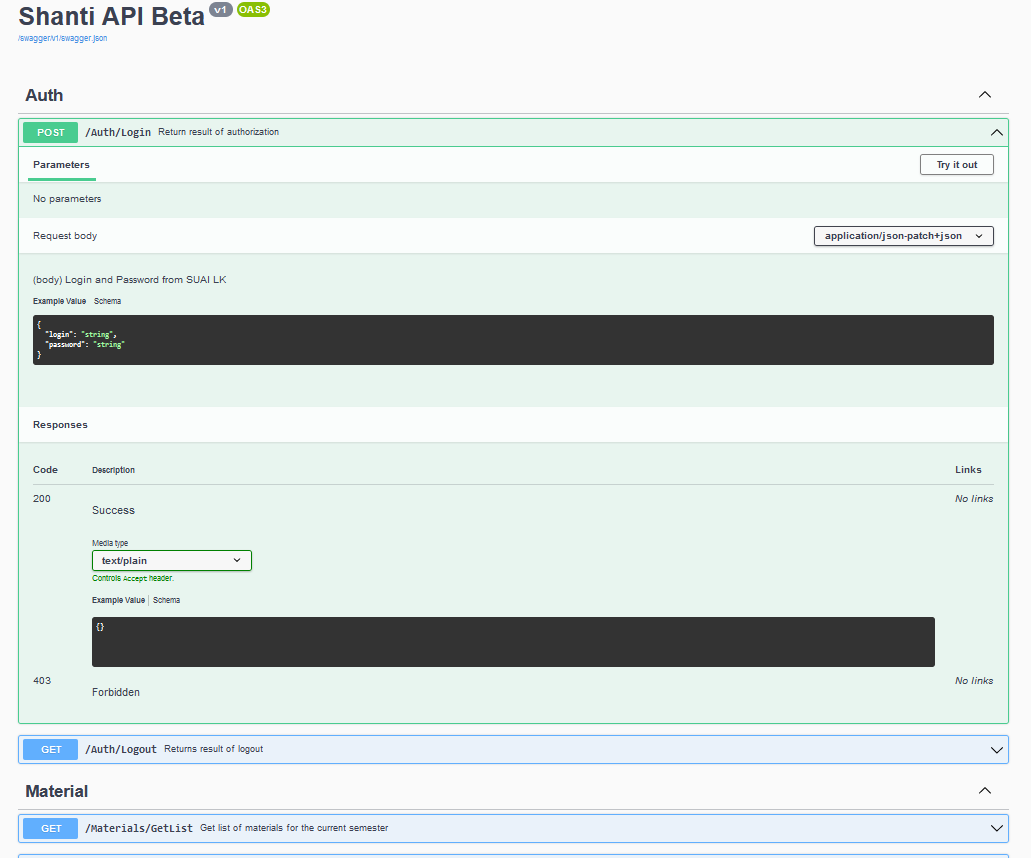
1. Была добавлена собственная система аутентификации на основе стандартных библиотек.

Для аутентификации используются те же данные для входа, что и в личном кабинете ГУАП, если вход в лк ГУАП совершен успешно – пользователь так же авторизуется в данном API и ему присваиваются необходимый cookies, на основании которого система уже будет автоматически подставлять необходимые для работы с лк ГУАП cookies.

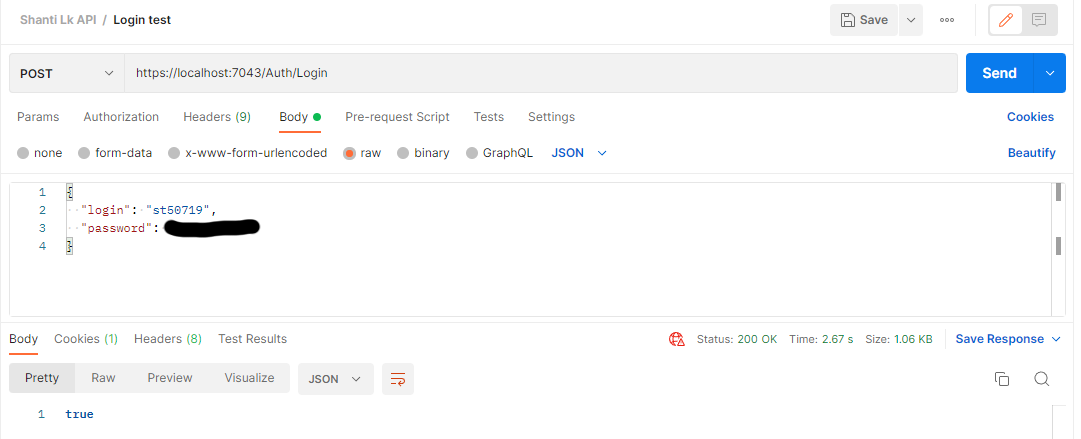
1. Все данные получаемые от лк ГУАП были выделены в отдельные классы с префиксом “s\_”, к ним прописаны правила JSON парсинга для приведения всех данных к единому стилю и дальнейшему их использованию в системе.
2. Все данные приходят и отправляются в формате JSON и в дальнейшем по необходимости переводятся в другие форматы для работы с лк ГУАП.
3. Документация была реализована с помощью системы автогенерации Swagger и открывается по умолчанию. В качестве исходных данных Swagger использует: атрибуты классов и методов, типы и названия параметров, xml аннотации к методам.
4. Все системные и неиспользуемые студентами данные отсеиваются на стадии формирования ответа в контроллерах.

# Полученные результаты

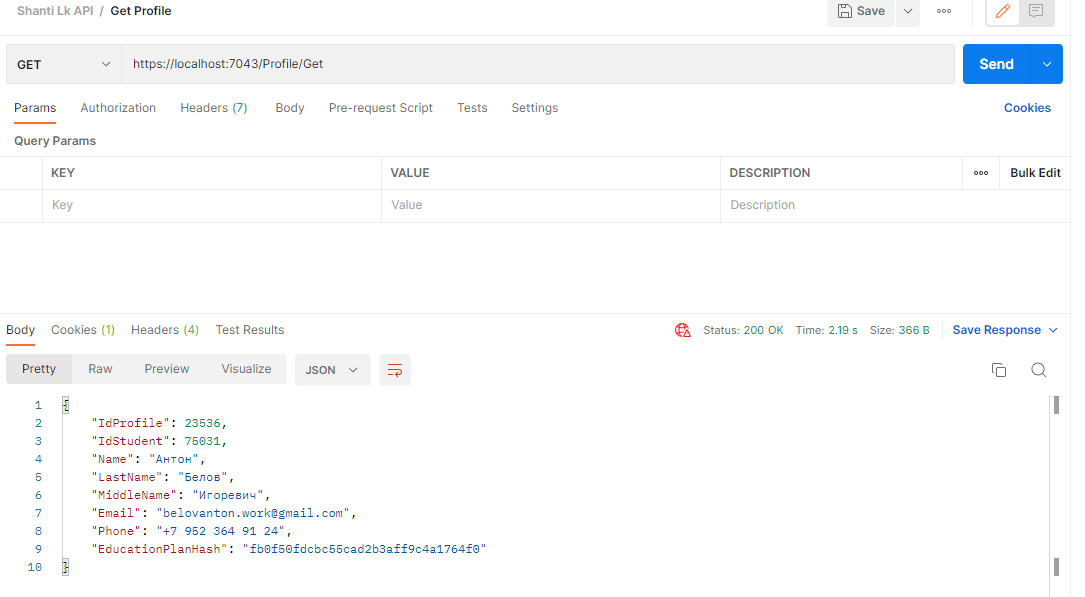
Страница по умолчанию (Swagger)



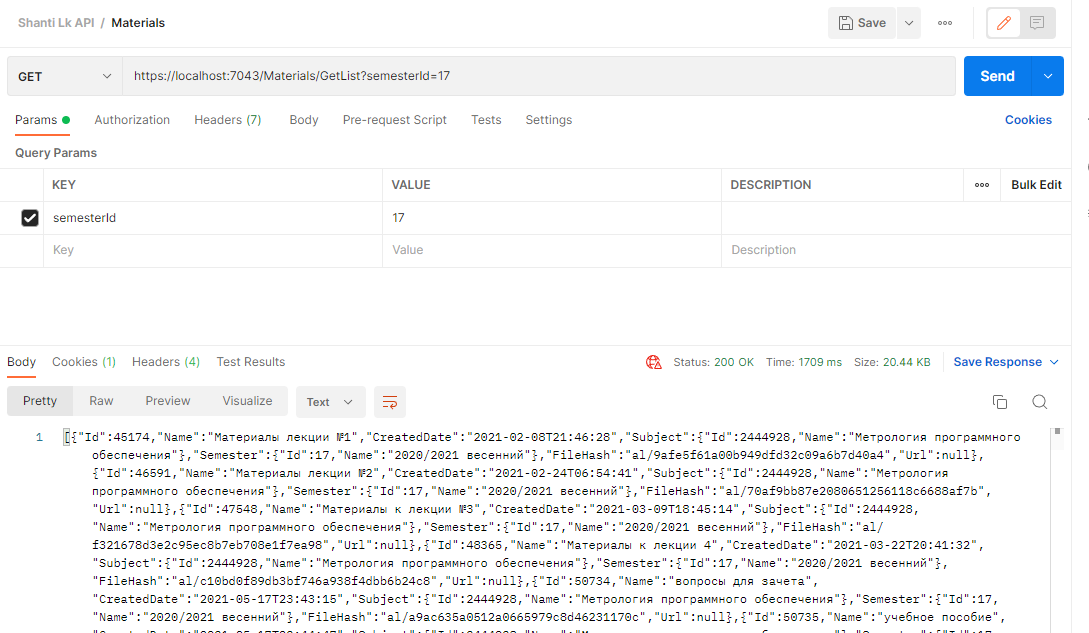
Аутентификация в системе



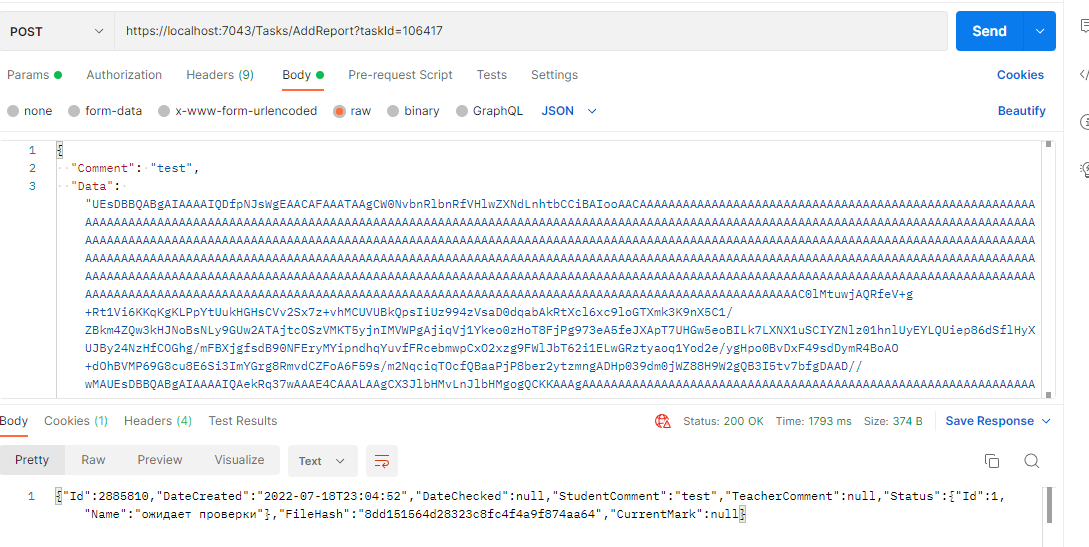
Основная информация профиля

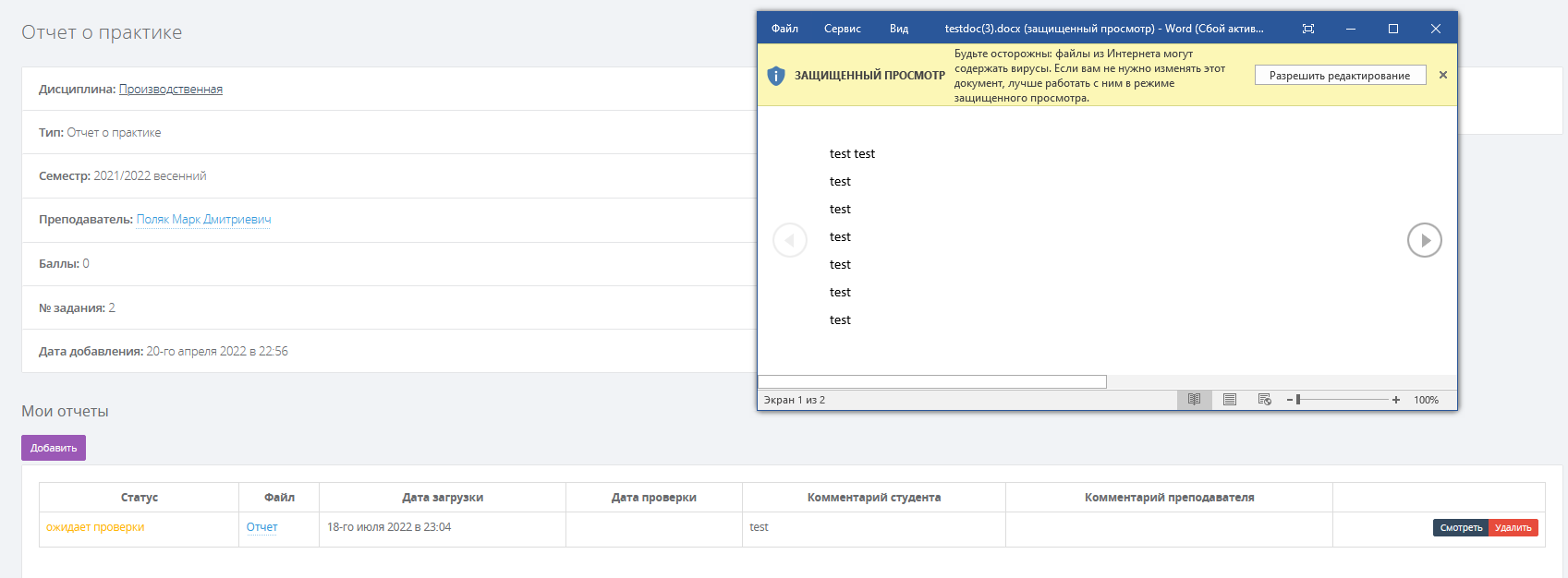


Список материалов за весенний семестр 2021



Загрузка нового отчета к заданию «Отчет о практике»





# Выводы

# Список использованной литературы

1. **C# 6.0 in a Nutshell**, Joseph Albahari, Ben Albahari.
2. **C# in Depth.** Jon Skeet.
3. ASP.NET Core. Разработка приложений. Джеймс Чамберс, Дэвид Пэкетт, Саймон Тиммс.
4. Проектирование веб-API. Лоре Арно
5. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. Марк Дж. Прайс

# Приложения

Листинг:

Startup.cs

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Mvc.Authorization;

using Microsoft.OpenApi.Models;

using System.Reflection;

namespace ShantiLk.Api

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddMvc(opt => { opt.EnableEndpointRouting = false; opt.Filters.Add(new AuthorizeFilter()); }).AddNewtonsoftJson();

services.AddControllers();

services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)

.AddCookie(options =>

{

options.Cookie.HttpOnly = true;

options.Cookie.SecurePolicy = CookieSecurePolicy.None;

options.Cookie.Name = "SimpleTalk.AuthCookieAspNetCore";

options.LoginPath = "/Auth/Login";

options.LogoutPath = "/Auth/Logout";

});

services.Configure<CookiePolicyOptions>(options =>

{

options.MinimumSameSitePolicy = SameSiteMode.Strict;

options.HttpOnly = Microsoft.AspNetCore.CookiePolicy.HttpOnlyPolicy.None;

options.Secure = CookieSecurePolicy.None;

});

services.AddEndpointsApiExplorer();

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "Shanti API Beta", Version = "v1" });

var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";

var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);

c.IncludeXmlComments(xmlPath, includeControllerXmlComments: true);

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Shanti API Beta");

c.RoutePrefix = String.Empty;

}

);

app.UseCookiePolicy();

app.UseAuthentication();

app.UseAuthorization();

app.UseMvc(opt =>

{

opt.MapRoute(

name: "default",

template: "{controller=Home}/{action=Index}/{codigo?}");

});

}

}

}

Programm.cs

namespace ShantiLk.Api

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

Appsettings.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

LaunchSettings.json

{

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:10341",

"sslPort": 44314

}

},

"profiles": {

"ShantiLk.Api": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"applicationUrl": "https://localhost:7043;http://localhost:5043",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

SuaiHttpClient.cs

sing System.Net;

using System.Security.Claims;

namespace ShantiLk.Api

{

public class SuaiHttpClient

{

private CookieContainer CookieContainer { get; set; }

private FormUrlEncodedContent FormUrlEncodedContent { get; set; }

private MultipartFormDataContent FormDataContent { get; set; }

private List<KeyValuePair<string, string>> EncodedValues { get; set; }

public SuaiHttpClient()

{

CookieContainer = new CookieContainer();

EncodedValues = new List<KeyValuePair<string, string>>();

FormDataContent = new MultipartFormDataContent();

}

public SuaiHttpClient(ClaimsPrincipal user)

{

CookieContainer = new CookieContainer();

var claims = user.Claims.Select(x => x.Value).ToList();

AddCookie("PHPSESSID", claims[1]);

AddCookie("sharedsessioID", claims[2]);

EncodedValues = new List<KeyValuePair<string, string>>();

FormDataContent = new MultipartFormDataContent();

}

public void AddCookie(string name, string value)

{

CookieContainer.Add(new Cookie(name, value, "/", ".guap.ru"));

}

public void AddFormData(string data, string key)

{

FormDataContent.Add(new StringContent(data), key);

}

public void AddFile(byte[] data, string key, string filename)

{

FormDataContent.Add(new ByteArrayContent(data), key, filename);

}

public void AddFormEncoded(string name, string value)

{

EncodedValues.Add(new KeyValuePair<string, string>(name, value));

}

private HttpClient CreateClient()

{

HttpClientHandler clientHandler = new HttpClientHandler();

clientHandler.AllowAutoRedirect = false;

if (CookieContainer.Count > 0)

clientHandler.CookieContainer = CookieContainer;

return new HttpClient(clientHandler);

}

public async Task<HttpResponseMessage> Get(string url)

{

HttpClient client = CreateClient();

HttpResponseMessage responce = await client.GetAsync(url);

if (responce.IsSuccessStatusCode || responce.StatusCode == HttpStatusCode.Found)

return responce;

else throw new Exception(responce.StatusCode.ToString());

}

public async Task<HttpResponseMessage> Post(string url)

{

HttpClient client = CreateClient();

FormUrlEncodedContent = new FormUrlEncodedContent(EncodedValues);

var req = new HttpRequestMessage(HttpMethod.Post, url) { Content = FormUrlEncodedContent };

HttpResponseMessage responce = await client.SendAsync(req);

if (responce.IsSuccessStatusCode || responce.StatusCode == HttpStatusCode.Found)

return responce;

else throw new Exception(responce.StatusCode.ToString());

}

public async Task<HttpResponseMessage> PostFile(string url)

{

HttpClient client = CreateClient();

var req = new HttpRequestMessage(HttpMethod.Post, url) { Content = this.FormDataContent };

HttpResponseMessage responce = await client.SendAsync(req);

if (responce.IsSuccessStatusCode || responce.StatusCode == HttpStatusCode.Found)

return responce;

else throw new Exception(responce.StatusCode.ToString());

}

public async Task<HttpResponseMessage> Delete(string url)

{

HttpClient client = CreateClient();

var req = new HttpRequestMessage(HttpMethod.Delete, url);

HttpResponseMessage responce = await client.SendAsync(req);

if (responce.IsSuccessStatusCode)

return responce;

else throw new Exception(responce.StatusCode.ToString());

}

}

}

s\_Teacher.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Teacher

{

public class s\_Teacher

{

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("username")]

public string Login { get; set; }

[JsonProperty("lastname")]

public string LastName { get; set; }

[JsonProperty("firstname")]

public string Name { get; set; }

[JsonProperty("middlename")]

public string MiddleName { get; set; }

[JsonProperty("email")]

public string Email { get; set; }

[JsonProperty("regEmail")]

public string RegistrationEmail { get; set; }

[JsonProperty("phone")]

public string Phone { get; set; }

[JsonProperty("image")]

public string ImageLink { get; set; }

[JsonProperty("degree\_name")]

public string Degree { get; set; }

[JsonProperty("auditorium")]

public string Auditorium { get; set; }

[JsonProperty("up\_contacts")]

public DateTime UpdateContactsDate { get; set; }

[JsonProperty("works")]

public List<s\_TeacherWork> Works { get; set; }

}

}

s\_Report.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Task

{

public class s\_Report

{

[JsonProperty("user\_fio")]

public string StudentName { get; set; }

[JsonProperty("prof\_fio")]

public string TeacherName { get; set; }

[JsonProperty("filelink")]

public string FileLink { get; set; }

[JsonProperty("status\_name")]

public string StatusName { get; set; }

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("task\_id")]

public string TaskId { get; set; }

[JsonProperty("task\_name")]

public string TaskName { get; set; }

[JsonProperty("task\_markpoint")]

public string IsReportRequired { get; set; }

[JsonProperty("status")]

public int StatusId { get; set; }

[JsonProperty("user\_id")]

public int StudentId { get; set; }

[JsonProperty("prof\_user")]

public int TeacherId { get; set; }

[JsonProperty("created")]

public DateTime? CreatedDate { get; set; }

[JsonProperty("checked")]

public DateTime? CheckedDate { get; set; }

[JsonProperty("stud\_comment")]

public string StudentComment { get; set; }

[JsonProperty("prof\_comment")]

public string TeacherComment { get; set; }

[JsonProperty("markpoint")]

public int? Mark { get; set; }

[JsonProperty("group\_id")]

public int GroupId { get; set; }

[JsonProperty("group\_num")]

public string GroupNumber { get; set; }

}

}

s\_Task.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Task

{

public class s\_Task

{

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("user\_id")]

public int TeacherId { get; set; }

[JsonProperty("datecreate")]

public DateTime? DateCreate { get; set; }

[JsonProperty("dateupdate")]

public DateTime? DateUpdate { get; set; }

[JsonProperty("harddeadline")]

public DateTime? DeadLine { get; set; }

[JsonProperty("description")]

public string[]? DescriptionArray { get; set; }

[JsonProperty("name")]

public string Name { get; set; }

[JsonProperty("semester")]

public int SemesterNumber { get; set; }

[JsonProperty("semester\_name")]

public string SemesterName { get; set; }

[JsonProperty("markpoint")]

public int? MaxMark { get; set; }

[JsonProperty("type")]

public int TaskTypeId { get; set; }

[JsonProperty("tt\_name")]

public string TaskTypeName { get; set; }

[JsonProperty("filelink")]

public string FileLink { get; set; }

[JsonProperty("filename")]

public string FileName { get; set; }

[JsonProperty("subject")]

public int SubjectId { get; set; }

[JsonProperty("subject\_name")]

public string SubjectName { get; set; }

[JsonProperty("groups")]

public int[]? Groups { get; set; }

[JsonProperty("fio")]

public string TeacherName { get; set; }

[JsonProperty("reportRequired")]

public int IsReportRequired { get; set; }

}

}

s\_TaskListItem.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Task

{

public class s\_TaskListItem

{

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("user\_id")]

public int TeacherId { get; set; }

[JsonProperty("datecreate")]

public DateTime? DateCreate { get; set; }

[JsonProperty("dateupdate")]

public DateTime? DateUpdate { get; set; }

[JsonProperty("name")]

public string Name { get; set; }

[JsonProperty("description")]

public string Description { get; set; }

[JsonProperty("type")]

public int TaskTypeId { get; set; }

[JsonProperty("tt\_name")]

public string TaskTypeName { get; set; }

[JsonProperty("semester")]

public int SemesterNumber { get; set; }

[JsonProperty("markpoint")]

public int? MaxMark { get; set; }

[JsonProperty("curPoints")]

public int? CurrentMark { get; set; }

[JsonProperty("reportRequired")]

public int IsReportRequired { get; set; }

[JsonProperty("harddeadline")]

public DateTime? DeadLine { get; set; }

[JsonProperty("grid")]

public int GroupId { get; set; }

[JsonProperty("subject")]

public int SubjectId { get; set; }

[JsonProperty("subject\_name")]

public string SubjectName { get; set; }

[JsonProperty("semester\_name")]

public string SemesterName { get; set; }

[JsonProperty("status")]

public int? StatusId { get; set; }

[JsonProperty("status\_name")]

public string StatusName { get; set; }

}

}

s\_Subject.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Subject

{

public class s\_Subject

{

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("IDEP")]

public int IDEP { get; set; }

[JsonProperty("chair")]

public string DepartmentName { get; set; }

[JsonProperty("chairId")]

public int DepartmentId { get; set; }

[JsonProperty("subj")]

public string Name { get; set; }

[JsonProperty("FYear")]

public int FirstYear { get; set; }

[JsonProperty("SemNum")]

public int SemesterNumber { get; set; }

[JsonProperty("contrText")]

public string ControlTypeName { get; set; }

[JsonProperty("IDContr")]

public int ControlTypeId { get; set; }

[JsonProperty("IsPractice")]

public int IsPractice { get; set; }

[JsonProperty("WeeksInTerm")]

public int CountWeeks { get; set; }

[JsonProperty("realSemNum")]

public int RealSemesterNumber { get; set; }

[JsonProperty("rpdAnnotaion")]

public s\_SubjectAnnotation Annotation { get; set; }

[JsonProperty("year")]

public int Year { get; set; }

[JsonProperty("semester")]

public string SemesterName { get; set; }

[JsonProperty("numberhours")]

public string CountHours { get; set; }

[JsonProperty("sumMarks")]

public int CurrentMark { get; set; }

[JsonProperty("summark")]

public int MaxMark { get; set; }

[JsonProperty("noticeMeSenpai")]

public List<string> Messages { get; set; }

[JsonProperty("nameMark")]

public string Mark { get; set; }

[JsonProperty("tasks")]

public List<s\_SubjectTask> Tasks { get; set; }

[JsonProperty("materials")]

public List<s\_SubjectMaterial> Materials { get; set; }

}

}

s\_SubjectFile.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Subject

{

public class s\_SubjectFile

{

[JsonProperty("epfiles\_name")]

public string Name { get; set; }

[JsonProperty("epfiles\_origname")]

public string FileName { get; set; }

[JsonProperty("epfiles\_type")]

public int Type { get; set; }

[JsonProperty("epfiles\_hash")]

public string Hash { get; set; }

}

}

s\_SubjectMaterial.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Subject

{

public class s\_SubjectMaterial

{

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("name")]

public string Name { get; set; }

[JsonProperty("url")]

public string Url { get; set; }

[JsonProperty("downloadlink")]

public string FileLink { get; set; }

[JsonProperty("grid")]

public int GroupId { get; set; }

[JsonProperty("grnum")]

public string GroupName { get; set; }

}

}

s\_SubjectTask.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Subject

{

public class s\_SubjectTask

{

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("name")]

public string Name { get; set; }

[JsonProperty("markpoint")]

public int MaxMark { get; set; }

[JsonProperty("grid")]

public int GroupId { get; set; }

[JsonProperty("grnum")]

public string GroupName { get; set; }

[JsonProperty("executed")]

public int IsExecuted { get; set; }

[JsonProperty("status")]

public int StatusId { get; set; }

[JsonProperty("status\_name")]

public string StatusName { get; set; }

[JsonProperty("curPoints")]

public int CurrentMark { get; set; }

}

}

s\_User.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Profile

{

public class s\_User

{

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("username")]

public string Login { get; set; }

[JsonProperty("firstname")]

public string Name { get; set; }

[JsonProperty("lastname")]

public string LastName { get; set; }

[JsonProperty("middlename")]

public string MiddleName { get; set; }

[JsonProperty("email")]

public string Email { get; set; }

[JsonProperty("phone")]

public string Phone { get; set; }

[JsonProperty("grnum")]

public string Group { get; set; }

[JsonProperty("vaccination")]

public bool IsVaccination { get; set; }

[JsonProperty("image")]

public string ImageLink { get; set; }

}

}

s\_Studentinfo.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Profile

{

public class s\_StudentInfo

{

[JsonProperty("STID")]

public int Id { get; set; }

[JsonProperty("FirstName")]

public string Name { get; set; }

[JsonProperty("LastName")]

public string LastName { get; set; }

[JsonProperty("MiddleName")]

public string MiddleName { get; set; }

[JsonProperty("Born")]

public DateTime DateOfBirth { get; set; }

[JsonProperty("StudentCardNum")]

public string StudentCardNum { get; set; }

[JsonProperty("Chair")]

public string DepartmentName { get; set; }

[JsonProperty("depId")]

public string InstituteId { get; set; }

[JsonProperty("depName")]

public string InstituteName { get; set; }

[JsonProperty("grid")]

public int GroupId { get; set; }

[JsonProperty("grnum")]

public string GroupNumber { get; set; }

[JsonProperty("deanshort")]

public string DeanName { get; set; }

[JsonProperty("dep\_secretary")]

public string SecretaryName { get; set; }

[JsonProperty("spec\_id")]

public int SpecialityId { get; set; }

[JsonProperty("spec\_name")]

public string SpecialityName { get; set; }

[JsonProperty("spec\_code")]

public string SpecialityCode { get; set; }

[JsonProperty("eduform\_id")]

public string EducationFormId { get; set; }

[JsonProperty("eduform\_name")]

public string EducationFormName { get; set; }

[JsonProperty("edutype\_id")]

public string EducationTypeId { get; set; }

[JsonProperty("edutype\_name")]

public string EducationTypeName { get; set; }

[JsonProperty("status")]

public string Status { get; set; }

}

}

s\_Material.cs

using Newtonsoft.Json;

namespace ShantiLk.Api.Models.SuaiClasses.Profile

{

public class s\_StudentInfo

{

[JsonProperty("STID")]

public int Id { get; set; }

[JsonProperty("FirstName")]

public string Name { get; set; }

[JsonProperty("LastName")]

public string LastName { get; set; }

[JsonProperty("MiddleName")]

public string MiddleName { get; set; }

[JsonProperty("Born")]

public DateTime DateOfBirth { get; set; }

[JsonProperty("StudentCardNum")]

public string StudentCardNum { get; set; }

[JsonProperty("Chair")]

public string DepartmentName { get; set; }

[JsonProperty("depId")]

public string InstituteId { get; set; }

[JsonProperty("depName")]

public string InstituteName { get; set; }

[JsonProperty("grid")]

public int GroupId { get; set; }

[JsonProperty("grnum")]

public string GroupNumber { get; set; }

[JsonProperty("deanshort")]

public string DeanName { get; set; }

[JsonProperty("dep\_secretary")]

public string SecretaryName { get; set; }

[JsonProperty("spec\_id")]

public int SpecialityId { get; set; }

[JsonProperty("spec\_name")]

public string SpecialityName { get; set; }

[JsonProperty("spec\_code")]

public string SpecialityCode { get; set; }

[JsonProperty("eduform\_id")]

public string EducationFormId { get; set; }

[JsonProperty("eduform\_name")]

public string EducationFormName { get; set; }

[JsonProperty("edutype\_id")]

public string EducationTypeId { get; set; }

[JsonProperty("edutype\_name")]

public string EducationTypeName { get; set; }

[JsonProperty("status")]

public string Status { get; set; }

}

}

Report.cs

namespace ShantiLk.Api.Models.ShantiClasses.Task

{

public class Report

{

public int Id { get; set; }

public DateTime? DateCreated { get; set; }

public DateTime? DateChecked { get; set; }

public string StudentComment { get; set; }

public string TeacherComment { get; set; }

public Dict.DictTaskStatus Status { get; set; }

public string FileHash { get; set; }

public int? CurrentMark { get; set; }

}

}

Task.cs

using ShantiLk.Api.Models.ShantiClasses.Dict;

using ShantiLk.Api.Models.ShantiClasses.Material;

namespace ShantiLk.Api.Models.ShantiClasses.Task

{

public class Task

{

public int Id { get; set; }

public string Name { get; set; }

public DateTime? DeadLine { get; set; }

public int? MaxMark { get; set; }

public int? CurrentMark { get; set; }

public DictSubject Subject { get; set; }

public DictSemester Semester { get; set; }

public DictTaskStatus Status { get; set; }

public DictTaskType Type { get; set; }

public DictTeacher Teacher { get; set; }

public List<Report> Reports { get; set; }

public DictFile File { get; set; }

}

}

Subject.cs

using ShantiLk.Api.Models.ShantiClasses.Dict;

namespace ShantiLk.Api.Models.ShantiClasses.Subject

{

public class Subject

{

public int Id { get; set; }

public string Name { get; set; }

public DictDepartment Department { get; set;}

public DictControlType ControlType { get; set; }

public DictSemester Semester { get; set; }

public string Mark { get; set; }

public string CountHours { get; set; }

public string WorkProgrammHash { get; set; }

public string EducationPlanHash { get; set; }

public string AnnotationHash { get; set; }

public List<string> Messages { get; set; }

public int MaxPoints { get; set; }

public int CurrentPoints { get; set; }

public List<SubjectTask> Tasks { get; set; }

public List<SubjectMaterial> Materials { get; set; }

}

}

SubjectListItem.cs

using ShantiLk.Api.Models.ShantiClasses.Dict;

using ShantiLk.Api.Models.SuaiClasses.Dict;

namespace ShantiLk.Api.Models.ShantiClasses.Subject

{

public class SubjectListItem

{

public int Id { get; set; }

public string Name { get; set; }

public DictSemester Semester { get; set; }

public DictControlType ControlType { get; set; }

public int? CountMessages { get; set; }

public List<DictTeacher> Teachers { get; set; }

}

}

DictSemester.cs

namespace ShantiLk.Api.Models.ShantiClasses.Dict

{

public class DictSemester

{

public int Id { get; set; }

public string Name { get; set; }

}

}

LoginData.cs

namespace ShantiLk.Api.Models.ShantiClasses.Common

{

public class LoginData

{

public string Login { get; set; }

public string Password { get; set; }

}

}

AuthController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

using ShantiLk.Api.Models.ShantiClasses.Common;

namespace ShantiLk.Api.Controllers

{

[Controller]

[Route("/Auth")]

public partial class AuthController : ControllerBase

{

/// <summary>

/// Return result of authorization

/// </summary>

/// <param name="data">(body) Login and Password from SUAI LK</param>

/// <returns></returns>

[Route("Login")]

[AllowAnonymous]

[HttpPost]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult Login([FromBody] LoginData data)

{

try { return Content(JsonConvert.SerializeObject(h\_Login(data).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Returns result of logout

/// </summary>

/// <param></param>

/// <returns></returns>

[Route("Logout")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult Logout()

{

try { return Content(JsonConvert.SerializeObject(h\_Logout().Result)); }

catch (Exception ex) { return Forbid(); }

}

}

}

AuthController.Helper.cs

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authentication.Cookies;

using ShantiLk.Api.Models.ShantiClasses.Common;

using System.Security.Claims;

namespace ShantiLk.Api.Controllers

{

public partial class AuthController

{

private async Task<bool> h\_Login(LoginData data)

{

SuaiHttpClient client = new SuaiHttpClient();

var responce = await client.Get("https://pro.guap.ru/exters/");

string sessionid = responce.Headers.SingleOrDefault(header => header.Key == "Set-Cookie").Value.First().Split(';')[0].Substring(10);

try

{

sessionid = GetSessionId(data.Login, data.Password, sessionid).Result;

string sharedid = GetSharedId(sessionid).Result;

return Authorization(data, new CookieData { SessionId = sessionid, SharedId = sharedid }).Result;

} catch (Exception ex)

{

return false;

}

}

private async Task<bool> h\_Logout()

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

var responce = await client.Get("https://pro.guap.ru/user/logout");

if (responce.StatusCode == System.Net.HttpStatusCode.Found)

{

HttpContext.SignOutAsync();

return true;

} else return false;

}

private async Task<bool> h\_CheckLogin()

{

SuaiHttpClient client = new SuaiHttpClient();

client.AddCookie("PHPSESSID", Request.Cookies["SessionId"]);

client.AddCookie("sharedsessioID", Request.Cookies["SharedId"]);

var responce = await client.Get("https://pro.guap.ru/inside\_s");

if (responce.IsSuccessStatusCode)

{

string result = responce.Content.ReadAsStringAsync().Result;

int i = result.IndexOf("user\_id");

string value = result.Substring(i + 9, result.IndexOf('"', i + 9));

return true;

}

else return false;

}

private async Task<string> GetSessionId(string username, string password, string sessionID)

{

SuaiHttpClient client = new SuaiHttpClient();

client.AddCookie("PHPSESSID", sessionID);

client.AddFormEncoded("\_username", username);

client.AddFormEncoded("\_password", password);

var responce = await client.Post("https://pro.guap.ru/user/login\_check");

if (!responce.IsSuccessStatusCode && responce.StatusCode != System.Net.HttpStatusCode.Found)

throw new Exception();

var sessid = responce.Headers.SingleOrDefault(header => header.Key == "Set-Cookie").Value;

return sessid.First().Split(';')[0].Substring(10);

}

private async Task<string> GetSharedId(string sessId)

{

SuaiHttpClient client = new SuaiHttpClient();

client.AddCookie("PHPSESSID", sessId);

var responce = await client.Get("https://pro.guap.ru/login\_redirect");

if (!responce.IsSuccessStatusCode && responce.StatusCode != System.Net.HttpStatusCode.Found)

throw new Exception();

var sharedId = responce.Headers.SingleOrDefault(header => header.Key == "Set-Cookie").Value;

return sharedId.First().Split(';')[0].Substring(15);

}

private async Task<bool> Authorization(LoginData loginData, CookieData cookieData)

{

var claims = new List<Claim>

{

new Claim(ClaimTypes.Name, loginData.Login),

new Claim(ClaimTypes.UserData, cookieData.SessionId),

new Claim(ClaimTypes.Hash, cookieData.SharedId),

};

ClaimsIdentity id = new ClaimsIdentity(claims, "ApplicationCookie", ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);

try

{

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(id));

} catch (Exception ex)

{

return false;

}

return true;

}

}

}

MaterialController.cs

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

namespace ShantiLk.Api.Controllers

{

[Route("/Materials")]

public partial class MaterialController : Controller

{

/// <summary>

/// Get list of materials for the current semester

/// </summary>

/// <param name="semesterId">(optional) Id needed semester</param>

/// <param name="subjectId">(optional) Id needed discipline</param>

/// <returns></returns>

[Route("GetList")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetMaterials(int semesterId = 0, int subjectId = 0)

{

try { return Content(JsonConvert.SerializeObject(h\_GetMaterials(semesterId, subjectId).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Download material file

/// </summary>

/// <param name="materialHash">Hash material file</param>

/// <returns></returns>

[Route("GetFile")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetFile(string materialHash)

{

try { return Content(JsonConvert.SerializeObject(h\_GetFile(materialHash).Result)); }

catch (Exception ex) { return Forbid(); }

}

}

}

MaterialController.Helper.cs

using Newtonsoft.Json;

using ShantiLk.Api.Models.ShantiClasses.Dict;

using ShantiLk.Api.Models.ShantiClasses.Material;

using ShantiLk.Api.Models.SuaiClasses.Answers;

namespace ShantiLk.Api.Controllers

{

public partial class MaterialController

{

private async Task<List<Material>> h\_GetMaterials(int? SemesterId=0, int? SubjectId=0)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

client.AddFormEncoded("iduser", "0");

HttpResponseMessage resp = client.Post("https://pro.guap.ru/getstudentmaterialdictionaries/").Result;

string result = resp.Content.ReadAsStringAsync().Result;

s\_MaterialAnswer answer = JsonConvert.DeserializeObject<s\_MaterialAnswer>(result);

if (SemesterId != 0 || SubjectId != 0)

{

client.AddFormEncoded("semester", SemesterId.ToString());

client.AddFormEncoded("subject", SubjectId.ToString());

resp = client.Post("https://pro.guap.ru/getstudentmaterials/").Result;

answer.Materials = JsonConvert.DeserializeObject<s\_MaterialAnswer>(resp.Content.ReadAsStringAsync().Result).Materials;

}

return answer.Materials.Select(x => new Material()

{

Id = x.Id,

Name = x.Name,

CreatedDate = x.CreatedDate,

Url = x.Url,

FileHash = x.FileLink.Substring(11),

Semester = new DictSemester

{

Id = x.SemesterId,

Name = answer.Dictionares.Semesters.FirstOrDefault(y => y.Id == x.SemesterId).Name

},

Subject = new DictSubject()

{

Id = x.SubjectIdsArray[0],

Name = answer.Dictionares.Subjects.FirstOrDefault(y => y.Id == x.SubjectIdsArray[0]).Name

}

}).ToList();

}

private async Task<byte[]> h\_GetFile(string FileHash)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

HttpResponseMessage resp = client.Get("https://pro.guap.ru/get-material/"+FileHash).Result;

byte[] result = resp.Content.ReadAsByteArrayAsync().Result;

return result;

}

}

}

ProfileController.cs

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

namespace ShantiLk.Api.Controllers

{

[Controller]

[Route("/Profile")]

public partial class ProfileController : Controller

{

/// <summary>

/// Get profile info

/// </summary>

/// <returns></returns>

[Route("Get")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult Get()

{

try { return Content(JsonConvert.SerializeObject(h\_GetProfile().Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Download education plan file

/// </summary>

/// <returns></returns>

[Route("GetEducationPlan")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetEducationPlan()

{

try { return Content(JsonConvert.SerializeObject(h\_GetEducationPlan().Result)); }

catch (Exception ex) { return Forbid(); }

}

}

}

ProfileController.Helper.cs

using Newtonsoft.Json;

using ShantiLk.Api.Models.ShantiClasses.Profile;

using ShantiLk.Api.Models.SuaiClasses.Answers;

using System.Security.Claims;

namespace ShantiLk.Api.Controllers

{

public partial class ProfileController

{

private async Task<ProfileInfo> h\_GetProfile()

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

HttpResponseMessage resp = client.Get("https://pro.guap.ru/inside\_s").Result;

string result = resp.Content.ReadAsStringAsync().Result;

string profileid = string.Empty;

if (HttpContext.User.Claims.Where(x=>x.Type == ClaimTypes.SerialNumber).Any())

{

profileid = HttpContext.User.Claims.Where(x => x.Type == ClaimTypes.SerialNumber).First().Value;

} else

{

int i = result.IndexOf("user\_id") + 10;

int i2 = result.IndexOf(",", i) - 1;

profileid = result.Substring(i, i2 - i);

var claims = new List<Claim>

{

new Claim(ClaimTypes.SerialNumber, profileid)

};

var appIdentity = new ClaimsIdentity(claims);

HttpContext.User.AddIdentity(appIdentity);

}

resp = client.Get("https://pro.guap.ru/getstudentprofile/" + profileid).Result;

result = resp.Content.ReadAsStringAsync().Result;

s\_ProfileAnswer answer = JsonConvert.DeserializeObject<s\_ProfileAnswer>(result);

return new ProfileInfo

{

IdProfile = answer.User.Id,

IdStudent = answer.Student.Id,

Email = answer.User.Email,

Phone = answer.User.Phone,

Name = answer.User.Name,

MiddleName = answer.User.MiddleName,

LastName = answer.User.LastName,

EducationPlanHash = answer.EducationPlan.Hash

};

}

private async Task<byte[]> h\_GetEducationPlan()

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

string hash = h\_GetProfile().Result.EducationPlanHash;

HttpResponseMessage resp = client.Get("https://pro.guap.ru/get-student-eduplan/" + hash).Result;

byte[] result = resp.Content.ReadAsByteArrayAsync().Result;

return result;

}

}

}

SubjectController.cs

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

namespace ShantiLk.Api.Controllers

{

[Route("/Subjects")]

public partial class SubjectController : Controller

{

/// <summary>

/// Get list of disciplines for current semester

/// </summary>

/// <param name="controlTypeId">(optional) Id needed control type</param>

/// <param name="semesterId">(optional) Id needed semester</param>

/// <returns></returns>

[Route("GetList")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetList(int semesterId = 0, int controlTypeId = 0)

{

try { return Content(JsonConvert.SerializeObject(h\_GetSubjects(semesterId, controlTypeId).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Get discipline info

/// </summary>

/// <param name="id"> Id needed discipline</param>

/// <returns></returns>

[Route("Get")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetTask(int id)

{

try { return Content(JsonConvert.SerializeObject(h\_GetSubject(id).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Get list tasks for discipline

/// </summary>

/// <param name="subjectId"> Id needed discipline</param>

/// <returns></returns>

[Route("GetTasks")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetTasks(int subjectId)

{

try { return Content(JsonConvert.SerializeObject(h\_GetSubjectTasks(subjectId).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Download annotation file for discipline

/// </summary>

/// <param name="subjectId"> Id needed discipline</param>

/// <returns></returns>

[Route("GetAnnotation")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetAnnotation(int subjectId)

{

try { return Content(JsonConvert.SerializeObject(h\_GetAnnotation(subjectId).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Download education plan file for discipline

/// </summary>

/// <param name="subjectId"> Id needed discipline</param>

/// <returns></returns>

[Route("GetEducationPlan")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetEducationPlan(int subjectId)

{

try { return Content(JsonConvert.SerializeObject(h\_GetEducationPlan(subjectId).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Download work programm file for discipline

/// </summary>

/// <param name="subjectId"> Id needed discipline</param>

/// <returns></returns>

[Route("GetWorkProgramm")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetWorkProgramm(int subjectId)

{

try { return Content(JsonConvert.SerializeObject(h\_GetWorkProgramm(subjectId).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Get list materials for discipline

/// </summary>

/// <param name="id"> Id needed discipline</param>

/// <returns></returns>

[Route("GetMaterials")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetMaterials(int id)

{

try { return Content(JsonConvert.SerializeObject(h\_GetSubjectMaterials(id).Result)); }

catch (Exception ex) { return Forbid(); }

}

}

}

SubjectController.Helper.cs

using Newtonsoft.Json;

using ShantiLk.Api.Models.ShantiClasses.Dict;

using ShantiLk.Api.Models.ShantiClasses.Subject;

using ShantiLk.Api.Models.SuaiClasses.Answers;

using ShantiLk.Api.Models.SuaiClasses.Subject;

namespace ShantiLk.Api.Controllers

{

public partial class SubjectController

{

private async Task<List<SubjectListItem>> h\_GetSubjects(int SemesterId = 0, int ControlTypeId = 0)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

client.AddFormEncoded("iduser", "0");

if (SemesterId != 0 || ControlTypeId != 0)

{

client.AddFormEncoded("semester", SemesterId.ToString());

client.AddFormEncoded("controltype", ControlTypeId.ToString());

}

HttpResponseMessage resp = client.Post("https://pro.guap.ru/getsubjectsdictionaries/").Result;

string result = resp.Content.ReadAsStringAsync().Result;

s\_SubjectListAnswer answer = JsonConvert.DeserializeObject<s\_SubjectListAnswer>(result);

return answer.Subjects.Select(x => new SubjectListItem()

{

Id = x.Id,

Name = x.Name,

CountMessages = x.NewMessagesCount,

ControlType = new DictControlType

{

Id = x.ControlTypeId,

Name = x.ControlTypeName

},

Semester = new DictSemester()

{

Id = x.SemesterNumber,

Name = x.SemesterName

},

Teachers = x.Teachers.Select(y => new DictTeacher

{

Id = y.Id,

Name = y.Name,

LastName = y.LastName,

MiddleName = y.MiddleName

}).ToList()

}).ToList();

}

private async Task<Subject> h\_GetSubject(int SubjectId)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

client.AddFormEncoded("id", SubjectId.ToString());

client.AddFormEncoded("eid", HttpContext.User.Claims.Select(x => x.Value).ToList()[0].Substring(2));

HttpResponseMessage resp = client.Post("https://pro.guap.ru/subjectItemStudent/").Result;

string result = resp.Content.ReadAsStringAsync().Result;

s\_Subject answer = JsonConvert.DeserializeObject<s\_SubjectAnswer>(result).Subject;

return new Subject()

{

Id = answer.Id,

Name = answer.Name,

Department = new DictDepartment

{

Id = answer.DepartmentId,

Name = answer.DepartmentName

},

ControlType = new DictControlType

{

Id = answer.ControlTypeId,

Name = answer.ControlTypeName

},

Semester = new DictSemester

{

Id = 0,

Name = answer.SemesterName

},

MaxPoints = answer.MaxMark,

CurrentPoints = answer.CurrentMark,

WorkProgrammHash = answer.Annotation.WorkProgramm.Hash,

AnnotationHash = answer.Annotation.Annotation.Hash,

EducationPlanHash = answer.Annotation.EducationPlan.Hash,

Messages = answer.Messages,

Mark = answer.Mark,

Tasks = answer.Tasks.Select(y => new SubjectTask

{

Id = y.Id,

Name = y.Name,

CurrentMark = y.CurrentMark,

MaxMark = y.MaxMark,

Status = new DictTaskStatus

{

Id = y.StatusId,

Name = y.StatusName

},

Group = new DictGroup

{

Id = y.GroupId,

Name = y.GroupName

},

IsExecuted = y.IsExecuted > 0 ? true : false

}).ToList(),

Materials = answer.Materials.Select(y => new SubjectMaterial

{

Id = y.Id,

Name = y.Name,

Url = y.Url,

FileHash = y.FileLink.Substring(11),

}).ToList()

};

}

private async Task<List<SubjectTask>> h\_GetSubjectTasks(int SubjectId)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

client.AddFormEncoded("id", SubjectId.ToString());

client.AddFormEncoded("eid", HttpContext.User.Claims.Select(x => x.Value).ToList()[0].Substring(2));

HttpResponseMessage resp = client.Post("https://pro.guap.ru/subjectItemStudent/").Result;

string result = resp.Content.ReadAsStringAsync().Result;

List<s\_SubjectTask> answer = JsonConvert.DeserializeObject<s\_SubjectAnswer>(result).Subject.Tasks;

return answer.Select(y => new SubjectTask

{

Id = y.Id,

Name = y.Name,

CurrentMark = y.CurrentMark,

MaxMark = y.MaxMark,

Status = new DictTaskStatus

{

Id = y.StatusId,

Name = y.StatusName

},

Group = new DictGroup

{

Id = y.GroupId,

Name = y.GroupName

},

IsExecuted = y.IsExecuted > 0 ? true : false

}).ToList();

}

private async Task<List<SubjectMaterial>> h\_GetSubjectMaterials(int SubjectId)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

client.AddFormEncoded("id", SubjectId.ToString());

client.AddFormEncoded("eid", HttpContext.User.Claims.Select(x => x.Value).ToList()[0].Substring(2));

HttpResponseMessage resp = client.Post("https://pro.guap.ru/subjectItemStudent/").Result;

string result = resp.Content.ReadAsStringAsync().Result;

List<s\_SubjectMaterial> answer = JsonConvert.DeserializeObject<s\_SubjectAnswer>(result).Subject.Materials;

return answer.Select(y => new SubjectMaterial

{

Id = y.Id,

Name = y.Name,

Url = y.Url,

FileHash = y.FileLink.Substring(11),

}).ToList();

}

private async Task<byte[]> h\_GetAnnotation(int SubjectId)

{

Subject subject = this.h\_GetSubject(SubjectId).Result;

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

HttpResponseMessage resp = client.Post("https://pro.guap.ru/get-student-eduplan/" + subject.AnnotationHash).Result;

return resp.Content.ReadAsByteArrayAsync().Result;

}

private async Task<byte[]> h\_GetWorkProgramm(int SubjectId)

{

Subject subject = this.h\_GetSubject(SubjectId).Result;

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

HttpResponseMessage resp = client.Post("https://pro.guap.ru/get-student-eduplan/" + subject.WorkProgrammHash).Result;

return resp.Content.ReadAsByteArrayAsync().Result;

}

private async Task<byte[]> h\_GetEducationPlan(int SubjectId)

{

Subject subject = this.h\_GetSubject(SubjectId).Result;

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

HttpResponseMessage resp = client.Post("https://pro.guap.ru/get-student-eduplan/" + subject.EducationPlanHash).Result;

return resp.Content.ReadAsByteArrayAsync().Result;

}

}

}

TaskController.cs

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

using ShantiLk.Api.Models.ShantiClasses.Task;

namespace ShantiLk.Api.Controllers

{

[Route("/Tasks")]

public partial class TaskController : Controller

{

/// <summary>

/// Get list of tasks for current semester

/// </summary>

/// <param name="semesterId">(optional) Id needed semester</param>

/// <param name="statusId">(optional) Id needed status task</param>

/// <param name="subjectId">(optional) Id needed discipline</param>

/// <param name="typeId">(optional) Id needed type task</param>

/// <returns></returns>

[Route("GetTasks")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetTasks(int semesterId = 0, int subjectId = 0, int typeId = 0, int statusId = 0)

{

try { return Content(JsonConvert.SerializeObject(h\_GetTasks(semesterId, subjectId, typeId, statusId).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Get task info

/// </summary>

/// <param name="id">Id needed task</param>

/// <returns></returns>

[Route("GetTask")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetTask(int id)

{

try { return Content(JsonConvert.SerializeObject(h\_GetTask(id).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Download task material file

/// </summary>

/// <param name="subjectId">Id needed task</param>

/// <returns></returns>

[Route("GetMaterial")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetMaterial(int subjectId)

{

try { return Content(JsonConvert.SerializeObject(h\_GetMaterialForTask(subjectId).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Get list of reports for task

/// </summary>

/// <param name="subjectId">Id needed task</param>

/// <returns></returns>

[Route("GetReports")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetReports(int taskId)

{

try { return Content(JsonConvert.SerializeObject(h\_GetReportsForTask(taskId).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Download report file

/// </summary>

/// <param name="reportHash">Hash report hash</param>

/// <returns></returns>

[Route("GetReport")]

[HttpGet]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult GetReport(string reportHash)

{

try { return Content(JsonConvert.SerializeObject(h\_DownloadReport(reportHash).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Add new report for task

/// </summary>

/// <param name="taskId">Id needed task</param>

/// <param name="report">(body) Comment, file name, byte array file</param>

/// <returns></returns>

[Route("AddReport")]

[HttpPost]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult AddReport(int taskId, [FromBody] NewReport report)

{

try { return Content(JsonConvert.SerializeObject(h\_AddReport(taskId, report).Result)); }

catch (Exception ex) { return Forbid(); }

}

/// <summary>

/// Delete report

/// </summary>

/// <param name="reportId">Id needed report</param>

/// <returns></returns>

[Route("DeleteReport")]

[HttpDelete]

[ProducesResponseType(200, Type = typeof(ActionResult))]

[ProducesResponseType(403, Type = typeof(void))]

public ActionResult DeleteReport(int reportId)

{

try { return Content(JsonConvert.SerializeObject(h\_DeleteReport(reportId).Result)); }

catch (Exception ex) { return Forbid(); }

}

}

}

TaskController.Helper.cs

using Newtonsoft.Json;

using ShantiLk.Api.Models.ShantiClasses.Dict;

using ShantiLk.Api.Models.ShantiClasses.Task;

using ShantiLk.Api.Models.SuaiClasses.Answers;

using ShantiLk.Api.Models.SuaiClasses.Task;

using Task = ShantiLk.Api.Models.ShantiClasses.Task.Task;

namespace ShantiLk.Api.Controllers

{

public partial class TaskController

{

private async Task<List<TaskListItem>> h\_GetTasks(int SemesterId, int SubjectId, int TypeId, int StatusId)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

client.AddFormEncoded("iduser", "0");

HttpResponseMessage resp = new HttpResponseMessage();

if (SemesterId != 0 || SubjectId != 0 || StatusId !=0 || TypeId !=0)

{

client.AddFormEncoded("semester", SemesterId.ToString());

client.AddFormEncoded("subject", SubjectId.ToString());

client.AddFormEncoded("type", TypeId.ToString());

client.AddFormEncoded("status", StatusId.ToString());

resp = client.Post("https://pro.guap.ru/get-student-tasks/").Result;

} else

{

resp = client.Post("https://pro.guap.ru/get-student-tasksdictionaries/").Result;

}

string result = resp.Content.ReadAsStringAsync().Result;

List<s\_TaskListItem> data = JsonConvert.DeserializeObject<s\_TaskListAnswer>(result).Tasks;

return data.Select(x => new TaskListItem()

{

Id = x.Id,

Name = x.Name,

DeadLine = x.DeadLine,

CurrentMark = x.CurrentMark,

MaxMark = x.MaxMark,

Type = new DictTaskType()

{

Id = x.TaskTypeId,

Name = x.TaskTypeName

},

Subject = new DictSubject()

{

Id = x.SubjectId,

Name = x.SubjectName

},

Semester = new DictSemester()

{

Id = x.SemesterNumber,

Name = x.SemesterName

},

Status = new DictTaskStatus()

{

Id = x.StatusId,

Name = x.StatusName

}

}).ToList();

}

private async Task<Task> h\_GetTask(int id)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

client.AddFormEncoded("task\_id", id.ToString());

HttpResponseMessage resp = client.Post("https://pro.guap.ru/get-student-task/" + id.ToString()).Result;

string result = resp.Content.ReadAsStringAsync().Result;

s\_TaskAnswer answer = JsonConvert.DeserializeObject<s\_TaskAnswer>(result);

s\_Task taskData = answer.TaskArray[0];

return new Task()

{

Id = taskData.Id,

Name = taskData.Name,

DeadLine = taskData.DeadLine,

MaxMark = taskData.MaxMark,

Type = new DictTaskType()

{

Id = taskData.TaskTypeId,

Name = taskData.TaskTypeName

},

Subject = new DictSubject()

{

Id = taskData.SubjectId,

Name = taskData.SemesterName

},

Semester = new DictSemester()

{

Id = taskData.SemesterNumber,

Name = taskData.SemesterName

},

Teacher = new DictTeacher()

{

Id = taskData.TeacherId,

Name = taskData.TeacherName

},

File = new DictFile()

{

Hash = taskData.FileLink.Length > 11 ? taskData.FileLink.Substring(11) : string.Empty,

Name = taskData.FileName

},

Reports = answer.Reports.Select(x=>new Report

{

Id = x.Id,

DateCreated = x.CreatedDate,

DateChecked = x.CheckedDate,

StudentComment = x.StudentComment,

TeacherComment = x.TeacherComment,

FileHash = x.FileLink.Substring(12),

CurrentMark = answer.Reports.LastOrDefault()?.Mark,

Status = new DictTaskStatus

{

Id = x.StatusId,

Name = x.StatusName

},

}).ToList()

};

}

private async Task<byte[]> h\_GetMaterialForTask(int id)

{

Task task = this.h\_GetTask(id).Result;

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

HttpResponseMessage resp = client.Post("https://pro.guap.ru/get-task/" + task.File.Hash).Result;

return resp.Content.ReadAsByteArrayAsync().Result;

}

private async Task<List<Report>> h\_GetReportsForTask(int id)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

client.AddFormEncoded("task\_id", id.ToString());

HttpResponseMessage resp = client.Post("https://pro.guap.ru/get-student-task/" + id.ToString()).Result;

string result = resp.Content.ReadAsStringAsync().Result;

s\_TaskAnswer answer = JsonConvert.DeserializeObject<s\_TaskAnswer>(result);

return answer.Reports.Select(x => new Report

{

Id = x.Id,

DateCreated = x.CreatedDate,

DateChecked = x.CheckedDate,

StudentComment = x.StudentComment,

TeacherComment = x.TeacherComment,

Status = new DictTaskStatus

{

Id = x.Id,

Name = x.StatusName

},

FileHash = x.FileLink.Substring(12),

CurrentMark = x.Mark

}).ToList();

}

private async Task<byte[]> h\_DownloadReport(string hash)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

HttpResponseMessage resp = client.Get("https://pro.guap.ru/get-report/" + hash).Result;

return resp.Content.ReadAsByteArrayAsync().Result;

}

private async Task<Report> h\_AddReport(int taskId, NewReport report)

{

Task task = this.h\_GetTask(taskId).Result;

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

client.AddFormData(taskId.ToString(), "task\_id");

client.AddFormData(report.Comment, "stud\_comment");

client.AddFile(report.Data, "file", report.FileName);

client.AddFormData(task.Teacher.Id.ToString(), "prof\_user");

client.AddFormData("1", "status");

HttpResponseMessage resp = client.PostFile("https://pro.guap.ru/reports").Result;

string answer = resp.Content.ReadAsStringAsync().Result;

s\_Report newReport = JsonConvert.DeserializeObject<s\_AddReportAnswer>(answer).NewReport;

return new Report

{

Id = newReport.Id,

StudentComment = newReport.StudentComment,

FileHash = newReport.FileLink.Substring(12),

Status = new DictTaskStatus

{

Id = newReport.StatusId,

Name = newReport.StatusName

},

DateCreated = newReport.CreatedDate

};

}

private async Task<bool> h\_DeleteReport(int reportId)

{

SuaiHttpClient client = new SuaiHttpClient(HttpContext.User);

HttpResponseMessage resp = client.Delete("https://pro.guap.ru/reports/"+reportId.ToString()).Result;

string answer = resp.Content.ReadAsStringAsync().Result;

return JsonConvert.DeserializeObject<s\_DeleteReportAnswer>(answer).Success == "success" ? true : false;

}

}

}