**Руководство программиста для работы с API**

**1. Введение**

Данное руководство предназначено для разработчиков, которые будут использовать наш API для интеграции с системой. API предоставляет доступ к основным функциям системы, таким как получение данных, управление ресурсами и выполнение операций.

**2. Базовые сведения**

**Базовый URL**  
Все запросы к API должны быть направлены на следующий базовый URL:

https://ezinxnacbigjfvsqhkqk.supabase.co

**3. Примеры запросов**

Авторизация:

try

{

var supabase = await SupabaseHelper.GetClient();

var response = await supabase

.From<User>()

.Where(x => x.Username == login && x.Password == password)

.Single();

if (response != null)

{

CurrentUserId = response.UserId;

OpenRoleSpecificForm(response.Role);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка авторизации: {ex.Message}");

}

Получение списка резюме:

try

{

var supabase = await SupabaseHelper.GetClient();

var resumes = await supabase

.From<Resume>()

.Get();

var table = new DataTable();

table.Columns.Add("ResumeId", typeof(int));

table.Columns.Add("Название", typeof(string));

table.Columns.Add("Должность", typeof(string));

table.Columns.Add("Зарплата", typeof(string));

table.Columns.Add("Образование", typeof(string));

table.Columns.Add("Навыки", typeof(string));

table.Columns.Add("График", typeof(string));

table.Columns.Add("Статус", typeof(string));

table.Columns.Add("Дата публикации", typeof(DateTime));

foreach (var resume in resumes.Models)

{

table.Rows.Add(

resume.ResumeId,

resume.Title,

resume.DesiredPosition,

resume.DesiredSalary,

resume.Education,

resume.ProfessionalSkills,

resume.WorkSchedule,

resume.Status,

resume.PublicationDate

);

}

dataGrid.DataSource = table;

dataGrid.Columns["ResumeId"].Visible = false;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке резюме: {ex.Message}");

}

Получение списка вакансий:

try

{

var supabase = await SupabaseHelper.GetClient();

var vacancies = await supabase

.From<Vacancy>()

.Get();

var companies = await supabase

.From<Company>()

.Get();

var table = new DataTable();

table.Columns.Add("VacancyId", typeof(int));

table.Columns.Add("Компания", typeof(string));

table.Columns.Add("Название вакансии", typeof(string));

table.Columns.Add("Специализация", typeof(string));

table.Columns.Add("Зарплата", typeof(string));

table.Columns.Add("Образование", typeof(string));

table.Columns.Add("График работы", typeof(string));

table.Columns.Add("Дата публикации", typeof(DateTime));

table.Columns.Add("Статус", typeof(string));

foreach (var vacancy in vacancies.Models)

{

var company = companies.Models.FirstOrDefault(c => c.CompanyId == vacancy.CompanyId);

string companyName = company?.CompanyName ?? "Неизвестная компания";

table.Rows.Add(

vacancy.VacancyId,

companyName,

vacancy.VacancyTitle,

vacancy.Specialization,

vacancy.Salary,

vacancy.Education,

vacancy.WorkSchedule,

vacancy.PublicationDate,

vacancy.Status

);

}

dataGrid.DataSource = table;

dataGrid.Columns["VacancyId"].Visible = false;

responseButton.Enabled = false;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке вакансий: {ex.Message}");

}

Получение информации о пользователе:

try

{

var supabase = await SupabaseHelper.GetClient();

var user = await supabase

.From<User>()

.Where(x => x.UserId == AuthForm.CurrentUserId)

.Single();

if (user != null)

{

string fullName = $"{user.LastName ?? ""} {user.FirstName ?? ""} {user.MiddleName ?? ""}".Trim();

if (string.IsNullOrEmpty(fullName))

fullName = "ФИО не указано";

labelFIO.Text = fullName;

labelUsername.Text = user.Username ?? "Не указано";

labelPassword.Text = user.Password ?? "Не указан";

labelPhone.Text = user.Phone ?? "Не указано";

labelEmail.Text = user.Email ?? "Не указано";

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки данных: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

Получение списка заявлений работодателя:

try

{

var supabase = await SupabaseHelper.GetClient();

var applications = await supabase

.From<EmployerApplication>()

.Where(x => x.UserId == AuthForm.CurrentUserId)

.Get();

var vacancies = await supabase

.From<Vacancy>()

.Get();

var table = new DataTable();

table.Columns.Add("ID заявки", typeof(int));

table.Columns.Add("Вакансия", typeof(string));

table.Columns.Add("Дата заявки", typeof(DateTime));

table.Columns.Add("Статус", typeof(string));

foreach (var application in applications.Models)

{

var vacancy = vacancies.Models.FirstOrDefault(v => v.VacancyId == application.VacancyId);

string vacancyTitle = vacancy?.VacancyTitle ?? "Неизвестная вакансия";

table.Rows.Add(

application.EmployerApplicationId,

vacancyTitle,

application.ApplicationDate,

application.ApplicationStatus

);

}

dataGrid.DataSource = table;

if (dataGrid.Columns.Contains("ID заявки"))

dataGrid.Columns["ID заявки"].Visible = false;

if (dataGrid.Columns.Contains("Дата заявки"))

dataGrid.Columns["Дата заявки"].DefaultCellStyle.Format = "dd.MM.yyyy HH:mm";

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке заявлений работодателя: {ex.Message}");

}

Получение списка заявлений соискателя:

try

{

var supabase = await SupabaseHelper.GetClient();

var applications = await supabase

.From<JobSeekerApplication>()

.Where(x => x.UserId == AuthForm.CurrentUserId)

.Get();

var resumes = await supabase

.From<Resume>()

.Where(r => r.UserId == AuthForm.CurrentUserId)

.Get();

var table = new DataTable();

table.Columns.Add("ID заявки", typeof(int));

table.Columns.Add("Резюме", typeof(string));

table.Columns.Add("Дата публикации", typeof(DateTime));

table.Columns.Add("Статус", typeof(string));

foreach (var application in applications.Models)

{

var resume = resumes.Models.FirstOrDefault(r => r.ResumeId == application.ResumeId);

string resumeTitle = resume?.Title ?? "Неизвестное резюме";

table.Rows.Add(

application.JobSeekerApplicationId,

resumeTitle,

application.ApplicationDate,

application.ApplicationStatus

);

}

dataGrid.DataSource = table;

if (dataGrid.Columns.Contains("ID заявки"))

dataGrid.Columns["ID заявки"].Visible = false;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке заявок: {ex.Message}");

}

Получение списка откликов:

try

{

var supabase = await SupabaseHelper.GetClient();

var responses = await supabase

.From<Response>()

.Where(x => x.UserId == AuthForm.CurrentUserId)

.Get();

var vacancies = await supabase

.From<Vacancy>()

.Get();

var companies = await supabase

.From<Company>()

.Get();

var table = new DataTable();

table.Columns.Add("ID", typeof(int));

table.Columns.Add("Вакансия", typeof(string));

table.Columns.Add("Компания", typeof(string));

table.Columns.Add("Дата отклика", typeof(DateTime));

foreach (var response in responses.Models)

{

var vacancy = vacancies.Models.FirstOrDefault(v => v.VacancyId == response.VacancyId);

if (vacancy != null)

{

var company = companies.Models.FirstOrDefault(c => c.CompanyId == vacancy.CompanyId);

string companyName = company?.CompanyName ?? "Неизвестная компания";

table.Rows.Add(

response.ResponseId,

vacancy.VacancyTitle,

companyName,

response.ResponseDate

);

}

}

dataGrid.DataSource = table;

dataGrid.Columns["ID"].Visible = false;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке откликов: {ex.Message}");

}

Получение списка приглашений:

try

{

var supabase = await SupabaseHelper.GetClient();

var invitations = await supabase

.From<Invitation>()

.Where(x => x.UserId == AuthForm.CurrentUserId)

.Get();

var resumes = await supabase

.From<Resume>()

.Get();

var companies = await supabase

.From<Company>()

.Get();

var table = new DataTable();

table.Columns.Add("ID", typeof(int));

table.Columns.Add("Резюме", typeof(string));

table.Columns.Add("Должность", typeof(string));

table.Columns.Add("Компания", typeof(string));

table.Columns.Add("Дата приглашения", typeof(DateTime));

table.Columns.Add("Статус", typeof(string));

foreach (var invitation in invitations.Models)

{

var resume = resumes.Models.FirstOrDefault(r => r.ResumeId == invitation.ResumeId);

if (resume != null)

{

var company = companies.Models.FirstOrDefault(c => c.CompanyId == invitation.UserId);

string companyName = company?.CompanyName ?? "Неизвестная компания";

table.Rows.Add(

invitation.InvitationId,

resume.Title,

resume.DesiredPosition,

companyName,

invitation.InvitationDate,

resume.Status

);

}

}

dataGrid.DataSource = table;

dataGrid.Columns["ID"].Visible = false;

dataGrid.Columns["Дата приглашения"].DefaultCellStyle.Format = "dd.MM.yyyy HH:mm";

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке приглашений: {ex.Message}");

}