**Frontend**

**Стек и сервисы**

**Blazorise**   
UI-Библиотека с готовыми стилями, иконками и шрифтами;

**Сервис для чтения/записи cookie**public interface ICookie

{

    public Task SetValueAsync(string key, string value, int? days = null);

    public Task<string> GetValueAsync(string key, string def = "");

}

**Сервис для установки заголовка Authorization и выхода из учётной записи**

public interface IAuth

{

    public Task<HttpStatusCode> Authorize();

    public Task Logout();

}

**Сервис для хранения данных пользователя**

public interface IAccount

{

    public AccountModel Current { get; set; }

    public bool IsEmpty { get; set; }

    public void Clear();

}

**Authorization.razor**

Взаимодействие пользователя с приложением начинается с экрана авторизации. В дальнейшем, если пользователь не авторизован или вышел из аккаунта, его отправит на эту страницу.

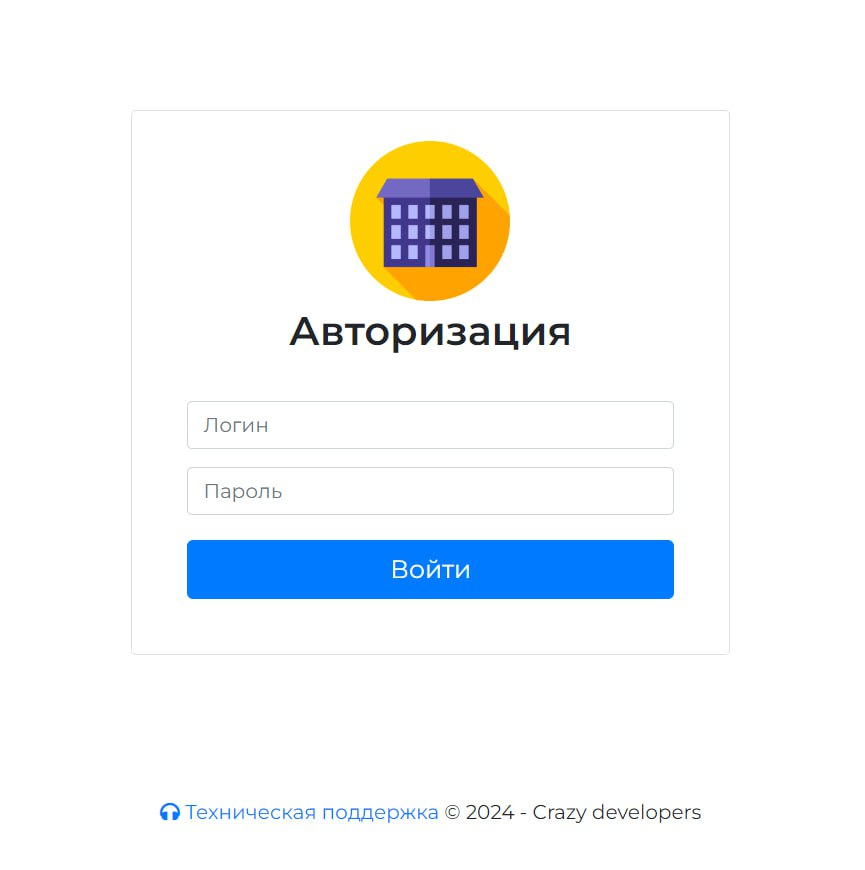
****

Рисунок 1 – Страница авторизации

async Task Submit()

{

Console.WriteLine("Submit started");

Console.WriteLine(JsonConvert.SerializeObject(authData));

var response = await httpClient.PostAsJsonAsync("http://localhost:6981/login", authData);

if (response.IsSuccessStatusCode)

{

var jsonResponse = await response.Content.ReadAsStringAsync();

AuthorizationModel? model = JsonConvert.DeserializeObject<AuthorizationModel>(jsonResponse);

await WriteTokenAsync(model);

Navigation.NavigateTo("/dashboard");

}

else

{

await wrongAuthAlert.Show();

return;

}

}

*– При нажатии на кнопку “Войти” отправляет post-запрос* [*/login*](http://localhost:6981/login) *с логином и паролем пользователя.*

**Dashboard.razor**

На домашней странице есть доступ ко всем функциям приложения.

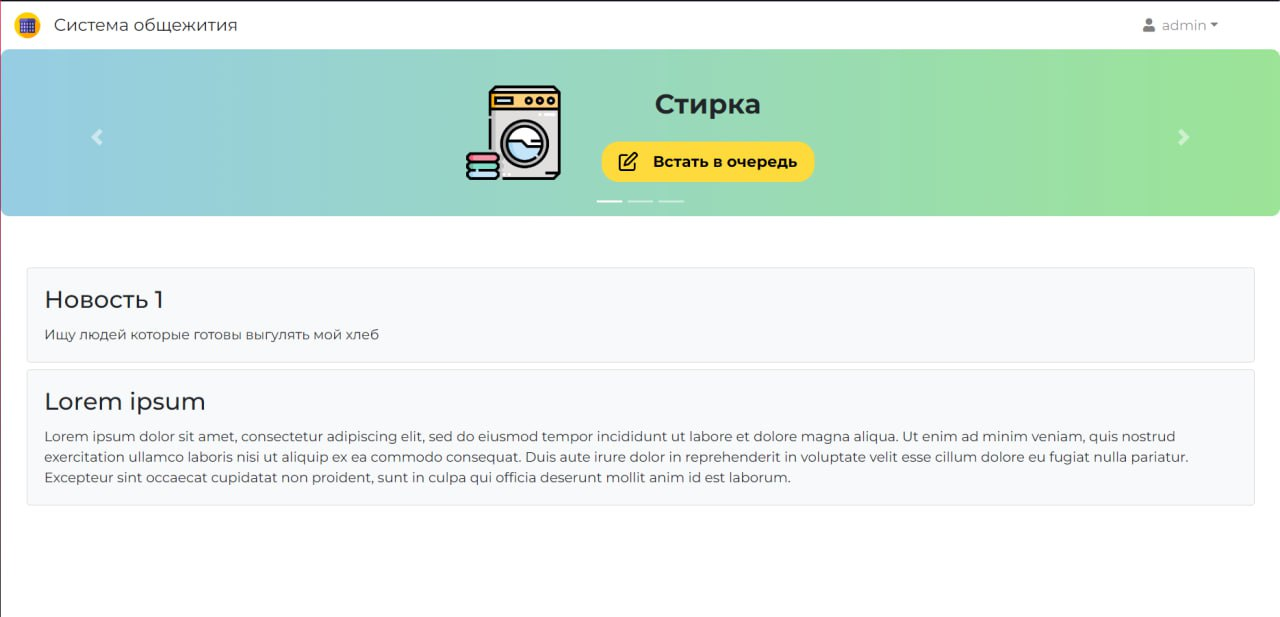


Рисунок 2.1-Главная страница



Рисунок 2.2

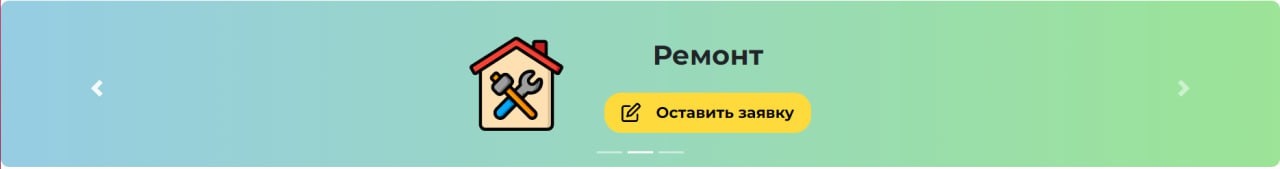


Рисунок 2.3

**Wash.razor**

На данной странице пользователь взаимодействует с очередью (может встать в нее или покинуть).

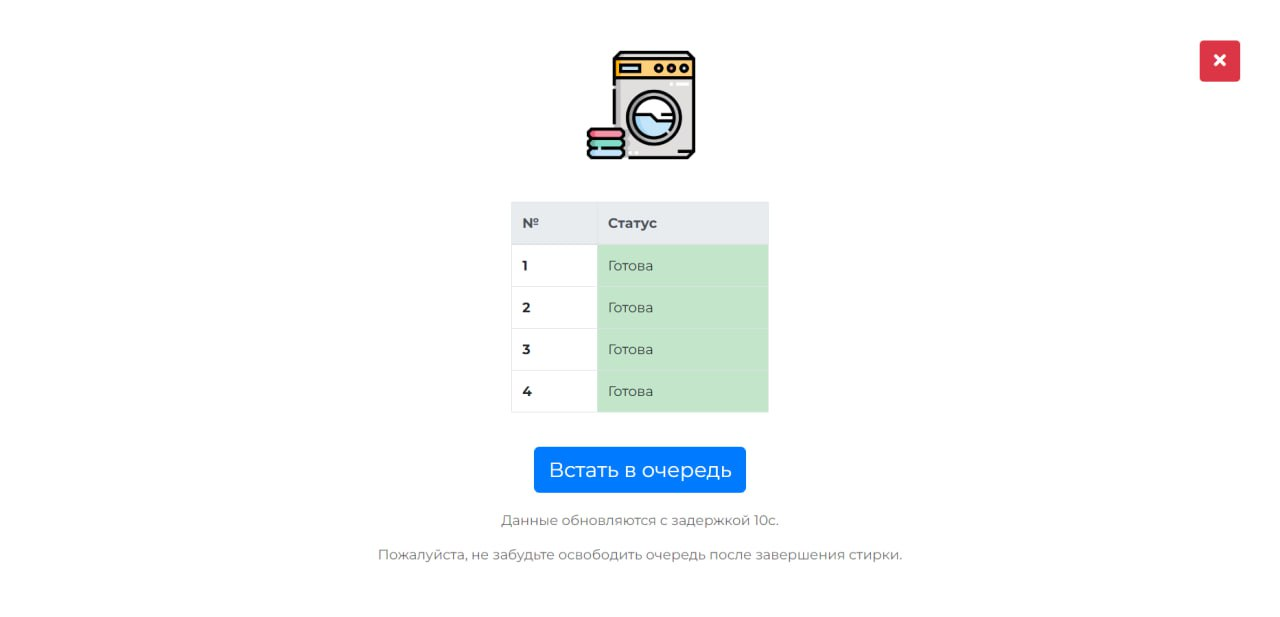


Рисунок 3.1-Wash.razor

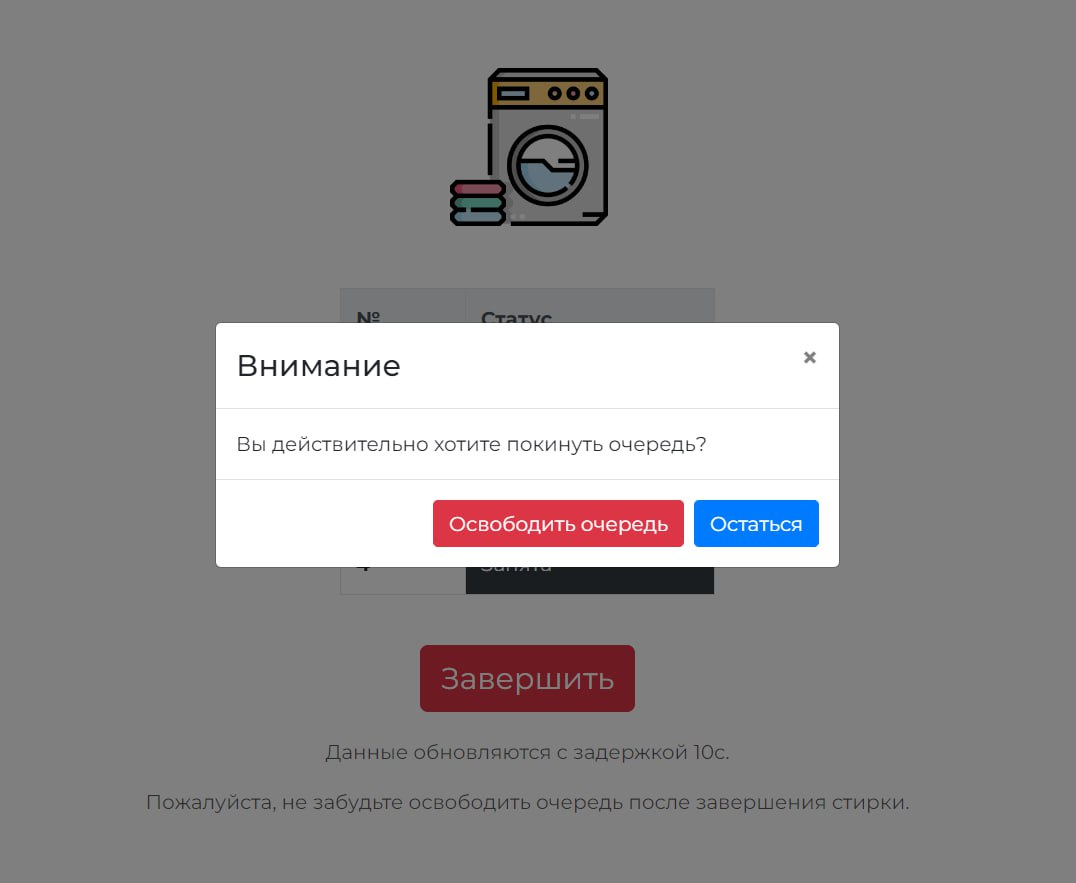


Рисунок 3.2-Wash.razor

async Task updateWashData()

{

if (await auth.Authorize() != HttpStatusCode.OK)

{

await auth.Logout();

return;

}

HttpResponseMessage laundryResponse = await httpClient.GetAsync("http://localhost:6981/laundry");

if (!laundryResponse.IsSuccessStatusCode)

{

await auth.Logout();

return;

}

string washersJson = await laundryResponse.Content.ReadAsStringAsync();

washers = JObject.Parse(washersJson)["washers"].Select(washer => new Washer((int)washer["id"], (int)washer["status"])).OrderBy(washer => washer.id).ToList();

Console.WriteLine(JsonConvert.SerializeObject(washers));

StateHasChanged();

}   
*– Каждые 10 секунд подгружает актуальные данные об очереди.*

async Task onAttemptToEnterQueue()

{

var httpResponse = await httpClient.PostAsync("http://localhost:6981/laundry", null);

Console.WriteLine(await httpResponse.Content.ReadAsStringAsync());

if (httpResponse.StatusCode == HttpStatusCode.Unauthorized)

{

await auth.Logout();

return;

}

QueueStatus = 1;

await updateWashData();

StateHasChanged();

}

async Task onAttemptToExitQueue()

{

var httpResponse = await httpClient.PutAsync("http://localhost:6981/laundry", null);

Console.WriteLine(await httpResponse.Content.ReadAsStringAsync());

if (httpResponse.StatusCode == HttpStatusCode.Unauthorized)

{

await auth.Logout();

return;

}

QueueStatus = 0;

await updateWashData();

StateHasChanged();

}  
*– Добавляет или удаляет пользователя из очереди.*