**Лабораторная работа №19**

**РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ.**

1. Демонстрация кода.

Листинг 1.1 – Сущность Master

namespace SUPIR\_RAZOR.Models.Entities

{

public class MasterEntity

{

public Guid Id { get; set; }

public string Name { get; set; } = string.Empty;

public string PhoneNumber { get; set; } = string.Empty;

public List<SupplyEntity> Supplies { get; set; } = [];

}

}

Листинг 1.2 – Конфигурация Master

namespace SUPIR\_RAZOR.Models.Configurations

{

public class MasterConfiguration : IEntityTypeConfiguration<MasterEntity>

{

public void Configure(EntityTypeBuilder<MasterEntity> builder)

{

builder.HasKey(m => m.Id);

builder.Property(m => m.Id)

.IsRequired()

.ValueGeneratedOnAdd();

builder.Property(m => m.Name)

.IsRequired()

.HasMaxLength(50);

builder.Property(m => m.PhoneNumber)

.IsRequired()

.HasMaxLength(12);

builder

.HasMany(m => m.Supplies)

.WithOne(s => s.Master)

.HasForeignKey(s => s.MasterId)

.OnDelete(DeleteBehavior.Restrict);

}

}

}

Листинг 1.3 –Репозиторий Master

using Microsoft.EntityFrameworkCore;

using SUPIR\_RAZOR.Models.Entities;

namespace SUPIR\_RAZOR.Data.Repositories

{

public class MastersRepository

{

private readonly SUPIR\_RAZORDbContext \_dbContext;

public MastersRepository(SUPIR\_RAZORDbContext dbContext)

{

\_dbContext = dbContext;

}

//Получение

public async Task<List<MasterEntity>> GetAll()

{

return await \_dbContext.Masters

.AsNoTracking()

.ToListAsync();

}

public async Task<List<MasterEntity>> GetAllNameSortASC()

{

return await \_dbContext.Masters

.AsNoTracking()

.OrderBy(m => m.Name)

.ToListAsync();

}

public async Task<List<MasterEntity>> GetAllNameSortDESC()

{

return await \_dbContext.Masters

.AsNoTracking()

.OrderByDescending(m => m.Name)

.ToListAsync();

}

public async Task<List<MasterEntity>> GetAllPhoneNumberSortASC()

{

return await \_dbContext.Masters

.AsNoTracking()

.OrderBy(m => m.PhoneNumber)

.ToListAsync();

}

public async Task<List<MasterEntity>> GetAllPhoneNumberSortDESC()

{

return await \_dbContext.Masters

.AsNoTracking()

.OrderByDescending(m => m.PhoneNumber)

.ToListAsync();

}

public async Task<List<MasterEntity>> GetWithSupplies()

{

return await \_dbContext.Masters

.AsNoTracking()

.Include(m => m.Supplies)

.ToListAsync();

}

public async Task<MasterEntity?> GetById(Guid id)

{

return await \_dbContext.Masters

.AsNoTracking()

.FirstOrDefaultAsync(m => m.Id == id);

}

public async Task<List<MasterEntity>> GetByName(String name)

{

return await \_dbContext.Masters

.AsNoTracking()

.Where(m => m.Name.Contains(name))

.ToListAsync();

}

public async Task<List<MasterEntity>> GetByPhoneNumber(string phoneNumber)

{

return await \_dbContext.Masters

.AsNoTracking()

.Where(m => m.PhoneNumber.Contains(phoneNumber))

.ToListAsync();

}

public async Task<List<MasterEntity>> GetByFilter(string name, string phoneNumber)

{

var query = \_dbContext.Masters.AsNoTracking();

if (!string.IsNullOrEmpty(name))

{

query = query.Where(m => m.Name.Contains(name));

}

if (!string.IsNullOrEmpty(phoneNumber))

{

query = query.Where(m => m.PhoneNumber.Contains(phoneNumber));

}

return await query.ToListAsync();

}

//Добавление

public async Task Add(Guid id, string name, string phoneNumber)

{

var masterEntity = new MasterEntity

{

Id = id,

Name = name,

PhoneNumber = phoneNumber

};

await \_dbContext.AddAsync(masterEntity);

await \_dbContext.SaveChangesAsync();

}

//Обновление

public async Task Update(MasterEntity master)

{

\_dbContext.Masters.Update(master);

await \_dbContext.SaveChangesAsync();

}

//Удаление

public async Task<DeleteResult> Delete(Guid id)

{

var master = await \_dbContext.Masters

.Include(m => m.Supplies)

.FirstOrDefaultAsync(m => m.Id == id);

if(master == null)

{

return DeleteResult.NotFound;

}

if(master.Supplies?.Any() == true)

{

return DeleteResult.HasDependencies;

}

\_dbContext.Masters.Remove(master);

await \_dbContext.SaveChangesAsync();

return DeleteResult.Success;

}

public enum DeleteResult

{

Success,

NotFound,

HasDependencies

}

}

}

Листинг 1.4 – Бекенд логика страницы Masters

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using SUPIR\_RAZOR.Data.Repositories;

using SUPIR\_RAZOR.Models.Entities;

using static SUPIR\_RAZOR.Data.Repositories.MastersRepository;

namespace SUPIR\_RAZOR.Pages

{

public class MastersModel : PageModel

{

private readonly MastersRepository \_repository;

public MastersModel(MastersRepository repository)

{

\_repository = repository;

}

[BindProperty]

public List<MasterEntity> Masters { get; set; } = new();

[BindProperty]

public MasterEntity NewMaster { get; set; } = new();

[BindProperty]

public MasterEntity UpdateMaster { get; set; } = new();

public async Task OnGetAsync()

{

Masters = await \_repository.GetAll();

}

public async Task OnGetSortByNameASCAsync()

{

Masters = await \_repository.GetAllNameSortASC();

}

public async Task OnGetSortByNameDESCAsync()

{

Masters = await \_repository.GetAllNameSortDESC();

}

public async Task OnGetSortByPhoneASCAsync()

{

Masters = await \_repository.GetAllPhoneNumberSortASC();

}

public async Task OnGetSortByPhoneDESCAsync()

{

Masters = await \_repository.GetAllPhoneNumberSortDESC();

}

public async Task OnGetSearchByIdAsync(Guid id)

{

var master = await \_repository.GetById(id);

Masters = master != null ? new List<MasterEntity> { master } : new List<MasterEntity>();

}

public async Task OnGetSearchByNameAsync(string name)

{

Masters = await \_repository.GetByName(name);

}

public async Task OnGetSearchByPhoneAsync(string phone)

{

Masters = await \_repository.GetByPhoneNumber(phone);

}

public async Task<IActionResult> OnPostDeleteAsync(Guid id)

{

var result = await \_repository.Delete(id);

switch (result)

{

case DeleteResult.HasDependencies:

TempData["ErrorMessage"] = "Cannot delete master with existing supplies";

break;

case DeleteResult.NotFound:

TempData["ErrorMessage"] = "Master not found";

break;

}

return RedirectToPage();

}

public async Task<IActionResult> OnPostAsync()

{

if (!ModelState.IsValid)

{

Masters = await \_repository.GetAll();

return RedirectToPage(); ;

}

await \_repository.Add(Guid.NewGuid(), NewMaster.Name, NewMaster.PhoneNumber);

return RedirectToPage();

}

public async Task<IActionResult> OnPostUpdateAsync(Guid id)

{

var existingMaster = await \_repository.GetById(id);

if(existingMaster == null)

{

ModelState.AddModelError("", "Master not found");

return RedirectToPage();

}

if (!string.IsNullOrEmpty(UpdateMaster.Name))

{

existingMaster.Name = UpdateMaster.Name;

}

if (!string.IsNullOrEmpty(UpdateMaster.PhoneNumber))

{

existingMaster.PhoneNumber = UpdateMaster.PhoneNumber;

}

await \_repository.Update(existingMaster);

return RedirectToPage();

}

}

}

Листинг 1.5 – Фронтенд логика страницы Masters

@page

@model SUPIR\_RAZOR.Pages.MastersModel

@{

ViewData["Title"] = "Masters";

}

<!-- Сортировка -->

<div class="sort-buttons">

<form method="get">

<button type="submit" name="handler" value="SortByNameASC">A-Z ↑</button>

<button type="submit" name="handler" value="SortByNameDESC">Z-A ↓</button>

<button type="submit" name="handler" value="SortByPhoneASC">Phone ↑</button>

<button type="submit" name="handler" value="SortByPhoneDESC">Phone ↓</button>

</form>

</div>

<!-- Поиск по ID -->

<form method="get">

<input type="text" name="id" placeholder="Enter ID" />

<button type="submit" name="handler" value="SearchById">Search by ID</button>

</form>

<!-- Поиск по имени -->

<form method="get">

<input type="text" name="name" placeholder="Enter Name" />

<button type="submit" name="handler" value="SearchByName">Search by Name</button>

</form>

<!-- Поиск по телефону -->

<form method="get">

<input type="text" name="phone" placeholder="Enter Phone" />

<button type="submit" name="handler" value="SearchByPhone">Search by Phone</button>

</form>

<!-- Форма добавления -->

<form method="post">

<div>

<label>

Name:

<input asp-for="NewMaster.Name" />

</label>

</div>

<div>

<label>

Phone:

<input asp-for="NewMaster.PhoneNumber" />

</label>

</div>

<button type="submit">Add Master</button>

</form>

<!-- Таблица с данными -->

<table>

<tr>

<th>ID</th>

<th>Name</th>

<th>Phone</th>

</tr>

@foreach (var master in Model.Masters)

{

<tr>

<td>@master.Id</td>

<td>@master.Name</td>

<td>@master.PhoneNumber</td>

<td>

<form method="post" asp-page-handler="Delete">

<input type="hidden" name="id" value="@master.Id" />

<button type="submit" onclick="return confirm('Delete this master?')">Delete</button>

</form>

</td>

</tr>

}

</table>

<!-- Форма обновления -->

<form method="post" asp-page-handler="Update">

<div>

<label>Master ID: <input type="text" name="id" required /></label>

</div>

<div>

<label>New Name (leave empty to keep current): <input asp-for="UpdateMaster.Name" /></label>

</div>

<div>

<label>New Phone (leave empty to keep current): <input asp-for="UpdateMaster.PhoneNumber" /></label>

</div>

<button type="submit">Update Master</button>

</form>

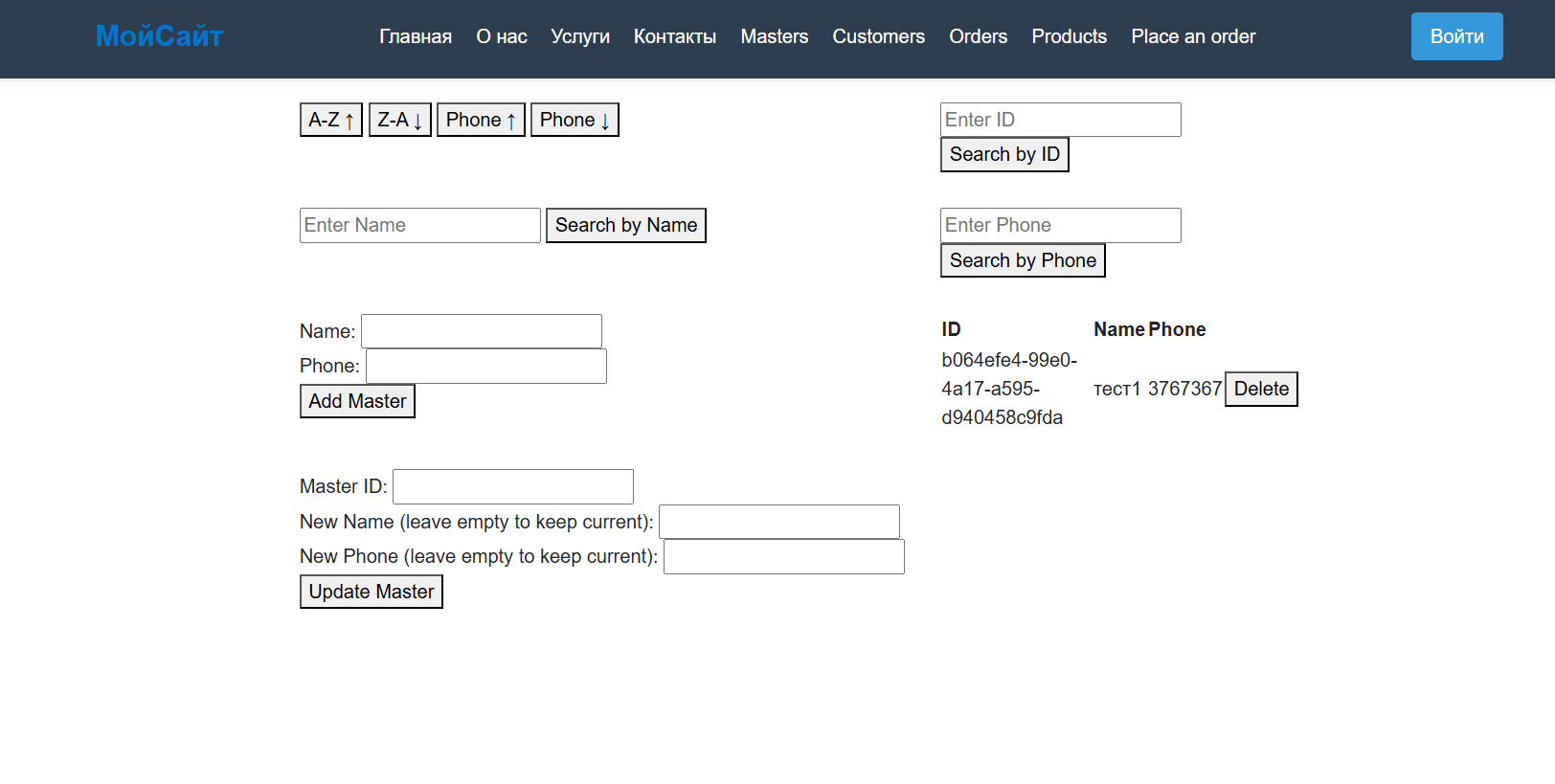


Рис. 1 – Страница работы с Masters.

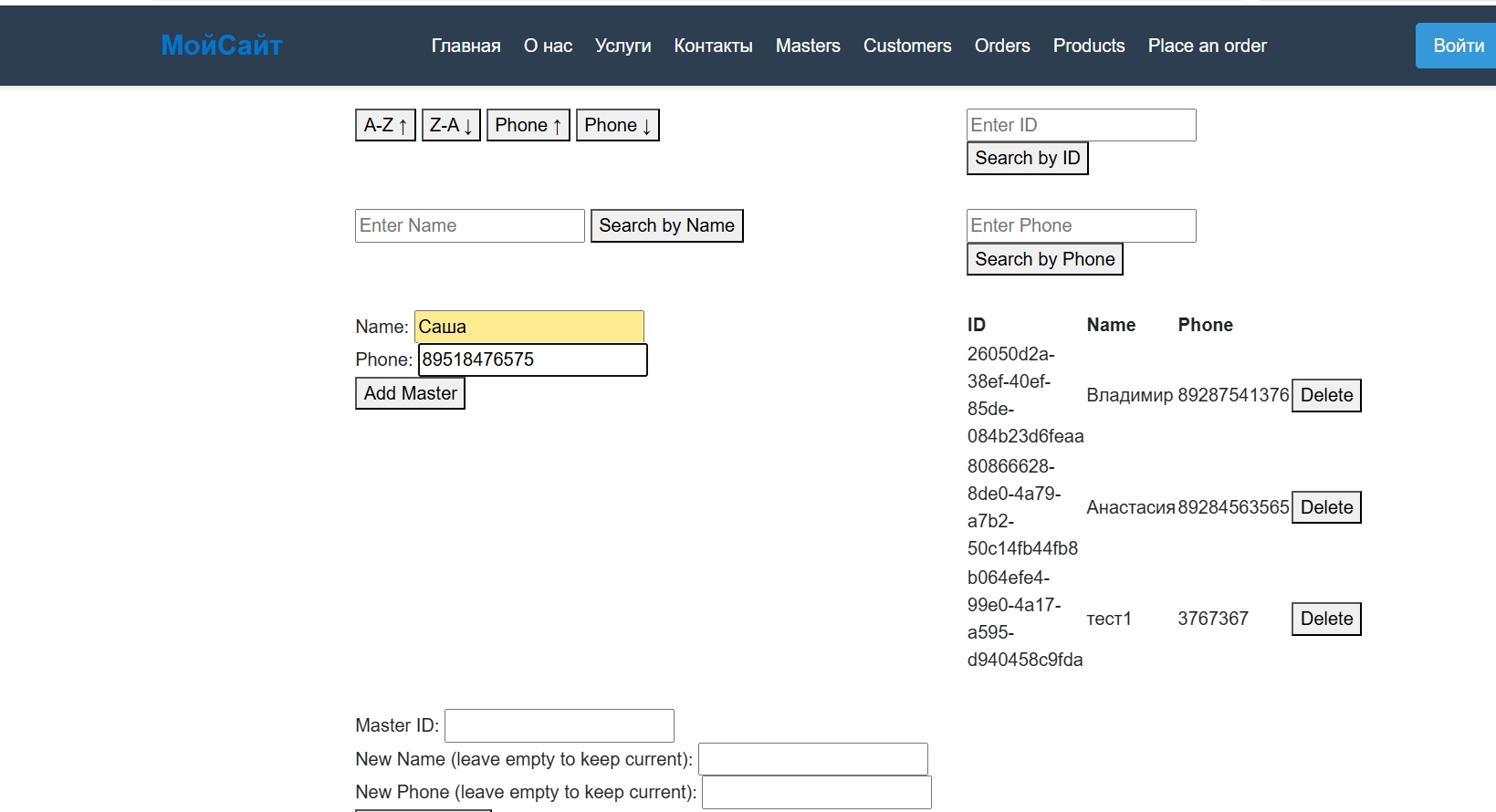




Рис. 2 – Добавление Master.

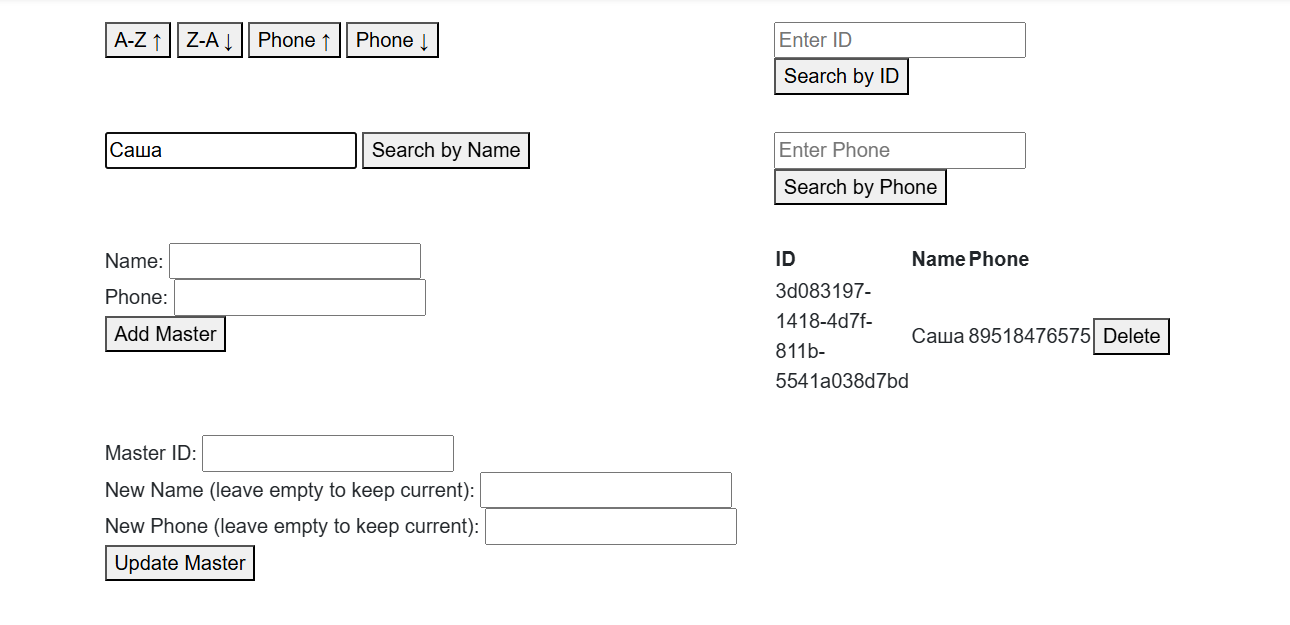


Рис. 3 – Поиск.

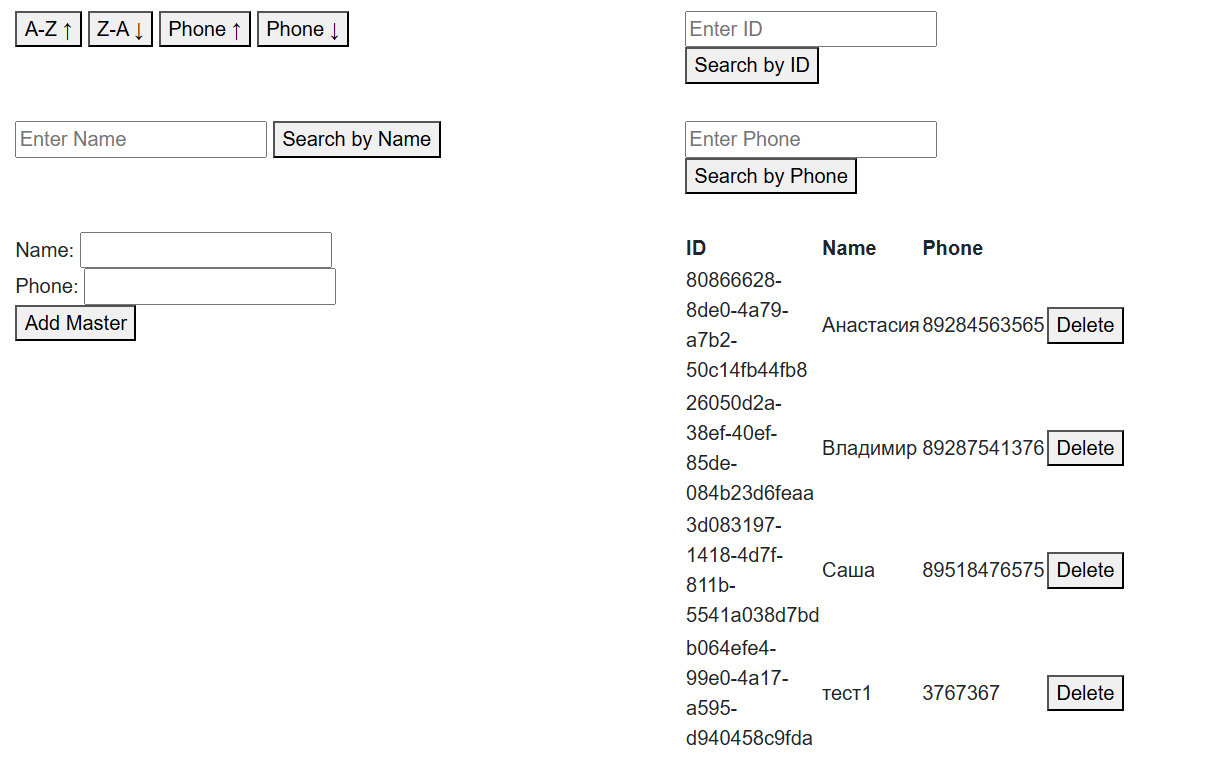
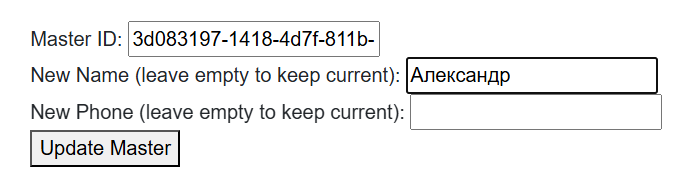


Рис. 4 –Сортировка.



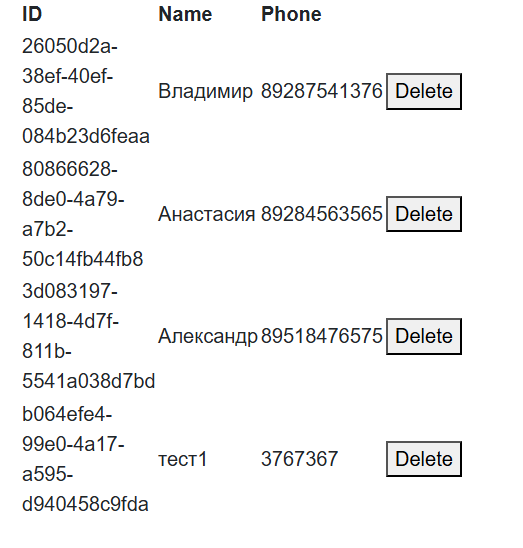


Рис. 5 – Обновление.

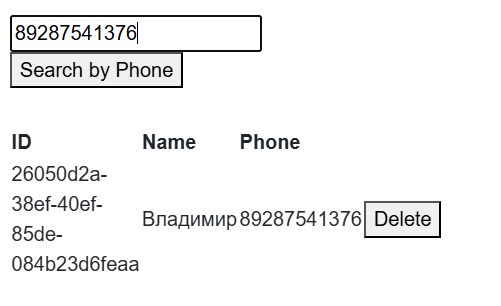
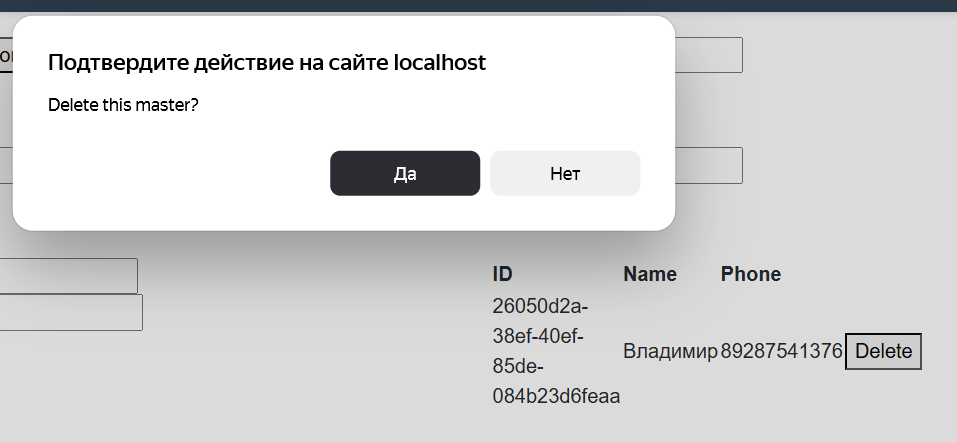


Рис. 6 – Поиск.



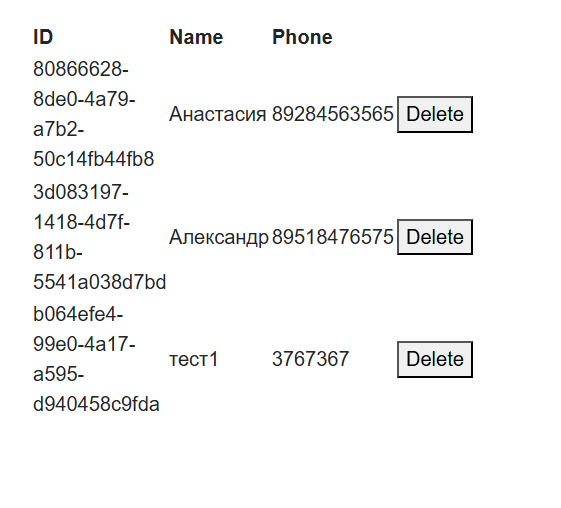


Рис. 6 – Удаление.