Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №4

по дисциплине «Технологии программирования»

Тема: «Web-сервисы (службы)»

Выполнил:

студент гр. ПРИ-123

А.Ц. Нямаа

Принял:

Ст. пр. кафедры ИСПИ

В.В. Данилов

Владимир, 2025 г.

ЦЕЛЬ РАБОТЫ:

Познакомиться с технологией создания и использования веб-сервисов.

ЗАДАНИЕ (вариант 12).

Создать веб-сервис, который возвращал бы данные, извлеченные из БД. Модифицировать веб-приложение, разработанное в предыдущей работе таким образом, чтобы оно получало данные из созданного веб-сервиса.

ВЫПОЛНЕНИЕ РАБОТЫ:

В начале выполнения данной работы я создал веб-сервис, возвращающий извлеченные из БД данные в формате JSON.

Листинг Program.cs:

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddRazorPages();

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

var app = builder.Build();

app.UseRouting();

app.MapRazorPages();

app.MapControllerRoute(

name: "default;",

pattern: "{controller=Home}/{action=Index}/{id?}"

);

app.Run();

Листинг ProductsController.cs:

using Microsoft.AspNetCore.Mvc;

using MySql.Data.MySqlClient;

using Api.RestApiClasses;

namespace Api.Constrollers

{

[Route("api/[controller]")]

[ApiController]

public class ProductsController : ControllerBase

{

string connection = "Server=localhost;database=task;user=root;password=1arthur\_nyamaa1";

// GET: api/<ValuesController>

[HttpGet]

public IEnumerable<Product> Get()

{

IList<Product> productSpecs = new List<Product>();

string sqlCommandProduct = "select \* from product";

using (MySqlConnection connectionCurrent = new MySqlConnection(connection))

{

connectionCurrent.Open();

MySqlCommand commandCurrent = new MySqlCommand(sqlCommandProduct, connectionCurrent);

using (var sqlSelectProduct = commandCurrent.ExecuteReader())

{

while (sqlSelectProduct.Read())

{

productSpecs.Add(new Product

{

ProductId = sqlSelectProduct.GetInt32("ProductId"),

ProductName = sqlSelectProduct.GetString("ProductName"),

ProductCost = sqlSelectProduct.GetInt32("ProductCost"),

ProductImgPath = sqlSelectProduct.GetString("ProductIMG")

});

}

}

}

return productSpecs;

}

// GET api/<ValuesController>/5

[HttpGet("{id}")]

public Product Get(int id)

{

Product productUnit = null;

string sqlCommandProductSingle = "select \* from product where ProductId = @idProductSingle";

using (MySqlConnection connectionCurrent = new MySqlConnection(connection))

{

connectionCurrent.Open();

MySqlCommand commandCurrent = new MySqlCommand(sqlCommandProductSingle, connectionCurrent);

commandCurrent.Parameters.AddWithValue("@idProductSingle", id);

using (var sqlSelectProduct = commandCurrent.ExecuteReader())

{

if (sqlSelectProduct.Read())

{

productUnit = new Product

{

ProductId = sqlSelectProduct.GetInt32("ProductId"),

ProductName = sqlSelectProduct.GetString("ProductName"),

ProductCost = sqlSelectProduct.GetInt32("ProductCost"),

ProductImgPath = sqlSelectProduct.GetString("ProductIMG")

};

}

};

};

return productUnit;

}

}

}

Листинг Product.cs:

namespace Api.RestApiClasses

{

public class Product

{

public int ProductId { get; set; }

public required string ProductName { get; set; }

public int ProductCost { get; set; }

public required string ProductImgPath { get; set; }

}

}

Листинг OrdersController.cs:

using Microsoft.AspNetCore.Mvc;

using MySql.Data.MySqlClient;

using Api.RestApiClasses;

namespace Api.Constrollers

{

[Route("api/[controller]")]

[ApiController]

public class OrdersController : ControllerBase

{

string connection = "Server=localhost;database=task;user=root;password=1arthur\_nyamaa1";

// GET: api/<OrderInfoController>

[HttpGet]

public IEnumerable<OrderInfo> Get()

{

IList<OrderInfo> OrderInfoSpecs = new List<OrderInfo>();

string sqlCommandProduct = "select \* from orderInfo";

using (MySqlConnection connectionCurrent = new MySqlConnection(connection))

{

connectionCurrent.Open();

MySqlCommand commandCurrent = new MySqlCommand(sqlCommandProduct, connectionCurrent);

using (var sqlSelectProduct = commandCurrent.ExecuteReader())

{

while (sqlSelectProduct.Read())

{

OrderInfoSpecs.Add(new OrderInfo

{

OrderId = sqlSelectProduct.GetInt32("OrderId"),

OrderName = sqlSelectProduct.GetString("OrderName"),

OrderSurname = sqlSelectProduct.GetString("OrderSurname"),

OrderEmail = sqlSelectProduct.GetString("OrderEmail"),

OrderQuantity = sqlSelectProduct.GetInt32("OrderQuantity"),

OrderAddress = sqlSelectProduct.GetString("OrderAddress")

});

}

}

}

return OrderInfoSpecs;

}

// GET api/<OrderInfoController>/5

[HttpGet("{id}")]

public OrderInfo Get(int id)

{

OrderInfo orderInfoUnit = null;

string sqlCommandProductSingle = "select \* from orderInfo where OrderId = @idOrder";

using (MySqlConnection connectionCurrent = new MySqlConnection(connection))

{

connectionCurrent.Open();

MySqlCommand commandCurrent = new MySqlCommand(sqlCommandProductSingle, connectionCurrent);

commandCurrent.Parameters.AddWithValue("@idOrder", id);

using (var sqlSelectProduct = commandCurrent.ExecuteReader())

{

if (sqlSelectProduct.Read())

{

orderInfoUnit = new OrderInfo

{

OrderId = sqlSelectProduct.GetInt32("OrderId"),

OrderName = sqlSelectProduct.GetString("OrderName"),

OrderSurname = sqlSelectProduct.GetString("OrderSurname"),

OrderEmail = sqlSelectProduct.GetString("OrderEmail"),

OrderQuantity = sqlSelectProduct.GetInt32("OrderQuantity"),

OrderAddress = sqlSelectProduct.GetString("OrderAddress")

};

}

};

};

return orderInfoUnit;

}

}

}

Листинг OrderInfo.cs:

namespace Api.RestApiClasses

{

public class OrderInfo

{

public int OrderId { get; set; }

public required string OrderName { get; set; }

public required string OrderSurname { get; set; }

public required string OrderEmail { get; set; }

public int OrderQuantity { get; set; }

public required string OrderAddress { get; set; }

}

}

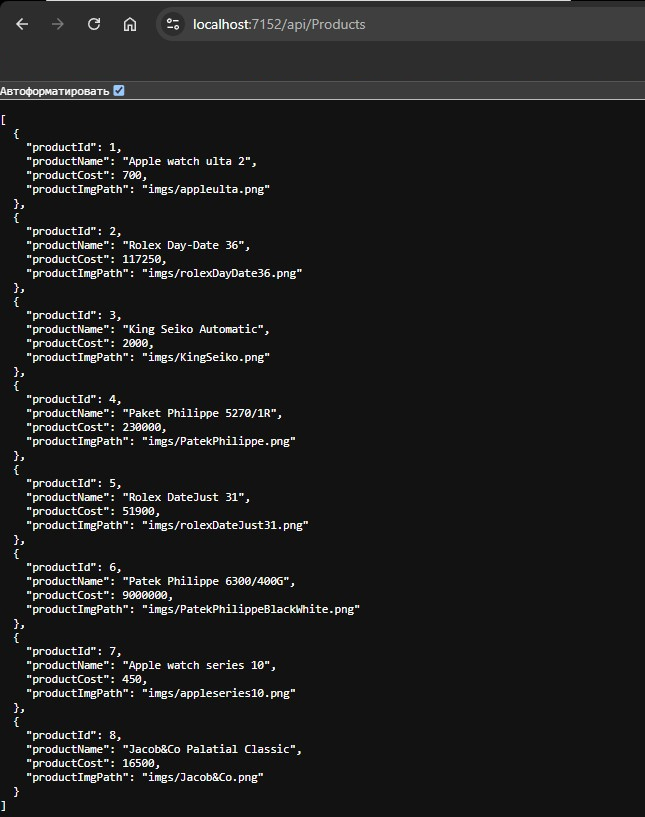


Рисунок 1. JSON-ответ, содержащий список продуктов.

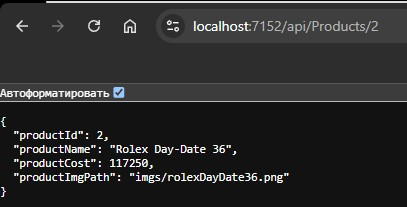


Рисунок 2. JSON-ответ, содержащий продукт определенного ID.

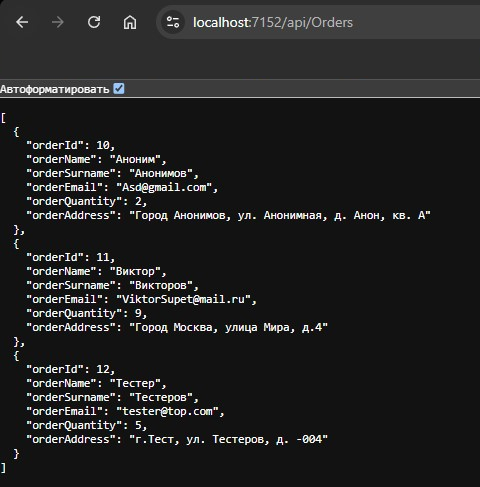


Рисунок 3. JSON-ответ, содержащий список заказов.

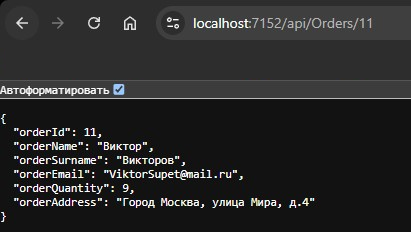


Рисунок 4. JSON-ответ, содержащий заказ определенного ID.

Модифицировал веб-приложение, разработанное в предыдущей работе таким образом, чтобы оно получало данные из созданного веб-сервиса.

Листинг Index.cshtml:

@page

@using calculator.Pages

@model calculator.Pages.IndexModel

@{

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Overpass:ital,wght@0,100..900;1,100..900&display=swap" rel="stylesheet">

<title>Clock Shop</title>

<link rel="stylesheet" href="~/css/StyleSheet.css">

</head>

<body>

<header>

<div class="leftText">

<h1> ДИЗАЙНЕРСКИЙ ЧАСЫ </h1>

<p> ПОЛНОСТЬЮ РУЧНОЕ ИЗГОТОВЛЕНИЕ </p>

<button class="buyHeader buttonToChange openModel" data-price="900000"> КУПИТЬ </button>

<dialog class="dialogMain">

<form class="orderForm" method="post">

<h2>Оформление заказа</h2>

@if (Model.successValid != "")

{

<p class="validSuccess">@Model.successValid</p>

}

else if (Model.errorValid != "")

{

<p class="validError">@Model.errorValid</p>

}

<p>

<label asp-for="order.NameOrder">Имя:</label>

<input type="text" id="name" asp-for="order.NameOrder" placeholder="Имя">

<span asp-validation-for="order.NameOrder" class="text-danger"></span>

</p>

<p>

<label asp-for="order.SecondNameOrder">Фамилия:</label>

<input type="text" id="SecondNameOrder" asp-for="order.SecondNameOrder" placeholder="Фамилия">

<span asp-validation-for="order.SecondNameOrder" class="text-danger"></span>

</p>

<p>

<label asp-for="order.EmailOrder">Почтовый ящик:</label>

<input type="email" id="email" asp-for="order.EmailOrder" placeholder="Почтовый ящик">

<span asp-validation-for="order.EmailOrder" class="text-danger"></span>

</p>

<p>

<label asp-for="order.AddressOrder">Адрес доставки:</label>

<input type="text" id="address" asp-for="order.AddressOrder" placeholder="Ул. Пушкино, д. 2, кв. 1">

<span asp-validation-for="order.AddressOrder" class="text-danger"></span>

</p>

<p>

<label asp-for="order.QuantityOrder">Количество:</label>

<input type="number" id="quantity" asp-for="order.QuantityOrder" placeholder="1">

<span asp-validation-for="order.QuantityOrder" class="text-danger"></span>

</p>

<p>

<label>Промокод:</label>

<input type="text" id="promo" placeholder="\*Промокод\*">

</p>

<p class="priceHeader"></p>

<button> Заказать! </button>

</form>

<button id="closeModel" class="closeModel">&#x2715 </button>

</dialog>

</div>

<div class="rightText">

<p> JACOB & CO <br>EPIC X TOURBILON BAGUETTE <br>900.000$</p>

<img src="imgs/mainPhotoWatches.png">

</div>

</header>

<main>

<div class="textInMain">

<p>ХИТЫ</p>

<p>ПОСЛЕДНИЕ ПОСТУПЛЕНИЯ</p>

</div>

<div class="productsList">

@foreach(var product in Model.products)

{

<div class="productUnit">

<img src="@product.imagePathProduct" />

<p>@product.NameProduct</p>

<p>@($"{product.CostProduct}$")</p>

<button class="buttonUnit buttonToChange openModel" data-price="@product.CostProduct"> Купить </button>

</div>

}

</div>

<div class="addModule orderContainer ">

<div class="textInMain addText textOrder">

<p>ЗАКАЗЫ</p>

</div>

<div class="orderItems">

<div class="orderItem">

<p class="singleItem">Id заказа</p>

<p class="singleItem">Имя покупателя</p>

<p class="singleItem">Фамилия покупателя</p>

<p class="singleItem">Почта покупателя</p>

<p class="singleItem"> Адресс доставки</p>

<p class="singleItem">Количество товара</p>

</div>

</div>

<div class="orderItems">

@foreach (var order in IndexModel.orders)

{

<div class="orderItem">

<p class="singleItem">@order.orderListId</p>

<p class="singleItem">@order.nameCustomer</p>

<p class="singleItem">@order.secondNameCustomer</p>

<p class="singleItem">@order.mailCustomer</p>

<p class="singleItem"> @order.addressCustomer</p>

<p class="singleItem">@order.quantityCustomer</p>

</div>

}

</div>

</div>

</main>

<script src="~/scripts/JavaScript.js"></script>

</body>

</html>

Листинг Index.cshtml.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using MySql.Data.MySqlClient;

using task2.Models;

namespace calculator.Pages

{

[IgnoreAntiforgeryToken]

public class IndexModel : PageModel

{

private HttpClient httpClient = new HttpClient();

private string apiUrlProducts = "https://localhost:7152/api/Products";

private string apiUrlOrders = "https://localhost:7152/api/Orders";

public List<Product> products = new List<Product>();

public static List<OrderList> orders = new List<OrderList>();

[BindProperty]

public Order order { get; set; } = new Order();

public async void OnGet()

{

products.Clear();

orders.Clear();

var responseGetProducts = httpClient.GetAsync(apiUrlProducts).Result;

if (responseGetProducts.IsSuccessStatusCode)

{

products = await responseGetProducts.Content.ReadFromJsonAsync<List<Product>>();

}

var responseGetOrders = httpClient.GetAsync(apiUrlOrders).Result;

if (responseGetOrders.IsSuccessStatusCode)

{

orders = await responseGetOrders.Content.ReadFromJsonAsync<List<OrderList>>();

}

}

public string successValid = "";

public string errorValid = "";

string connection = "Server=localhost;database=task;user=root;password=1arthur\_nyamaa1";

public void OnPost()

{

if (!ModelState.IsValid)

{

errorValid = "Некоторые данные были введены неправильно!";

return;

}

string sqlCommand = "insert into orderinfo (OrderName, OrderSurname, OrderEmail, OrderAddress, OrderQuantity)" +

"values (@NameOrder, @SecondNameOrder, @EmailOrder, @AddressOrder, @QuantityOrder); ";

using (MySqlConnection connectionCurrent = new MySqlConnection(connection))

{

connectionCurrent.Open();

MySqlCommand commandCurrent = new MySqlCommand(sqlCommand, connectionCurrent);

commandCurrent.Parameters.AddWithValue("@NameOrder", order.NameOrder);

commandCurrent.Parameters.AddWithValue("@SecondNameOrder", order.SecondNameOrder);

commandCurrent.Parameters.AddWithValue("@EmailOrder", order.EmailOrder);

commandCurrent.Parameters.AddWithValue("@AddressOrder", order.AddressOrder);

commandCurrent.Parameters.AddWithValue("@QuantityOrder", order.QuantityOrder);

commandCurrent.ExecuteNonQuery();

connectionCurrent.Close();

}

OnGet();

successValid = $"Спасибо {order.NameOrder} {order.SecondNameOrder}, заказ выполнен успешно!{Environment.NewLine}" +

$"Доставим товар по адресу {order.AddressOrder} в количестве {order.QuantityOrder} шт.{Environment.NewLine}" +

$"Электронный чек придет в ближайшее время на почту {order.EmailOrder}";

;

}

}

}

Листинг Product.cs:

using System.Text.Json.Serialization;

public class Product

{

[JsonPropertyName("productId")]

public int IdProduct { get; set; }

[JsonPropertyName("productName")]

public required string NameProduct { get; set; }

[JsonPropertyName("productCost")]

public int CostProduct { get; set; }

[JsonPropertyName("productImgPath")]

public required string imagePathProduct { get; set; }

}

Листинг Order.cs:

using System.ComponentModel.DataAnnotations;

using System.Text.Json.Serialization;

namespace task2.Models

{

public class Order

{

[Required(ErrorMessage = "Имя обязательно!")]

[StringLength(50, MinimumLength = 2, ErrorMessage = "Не менее 2 и не более 50 символов")]

public string NameOrder { get; set; }

[Required(ErrorMessage = "Фамилия обязательна!")]

[StringLength(50, MinimumLength = 2, ErrorMessage = "Не менее 2 и не более 50 символов")]

public string SecondNameOrder { get; set; }

[Required(ErrorMessage = "Почтовый ящик обязателен!")]

[EmailAddress(ErrorMessage = "Некорректый адрес")]

public string EmailOrder { get; set; }

[Required(ErrorMessage = "Адрес обязателен!")]

public string AddressOrder { get; set; }

[Required(ErrorMessage = "Количество обязательно!")]

[Range(1, 10, ErrorMessage = "Недопустимое количество")]

public int QuantityOrder { get; set; }

}

public class OrderList

{

[JsonPropertyName("orderId")]

public int orderListId { get; set; }

[JsonPropertyName("orderName")]

public required string nameCustomer { get; set; }

[JsonPropertyName("orderSurname")]

public required string secondNameCustomer { get; set; }

[JsonPropertyName("orderEmail")]

public required string mailCustomer { get; set; }

[JsonPropertyName("orderAddress")]

public required string addressCustomer { get; set; }

[JsonPropertyName("orderQuantity")]

public int quantityCustomer { get; set; }

}

}

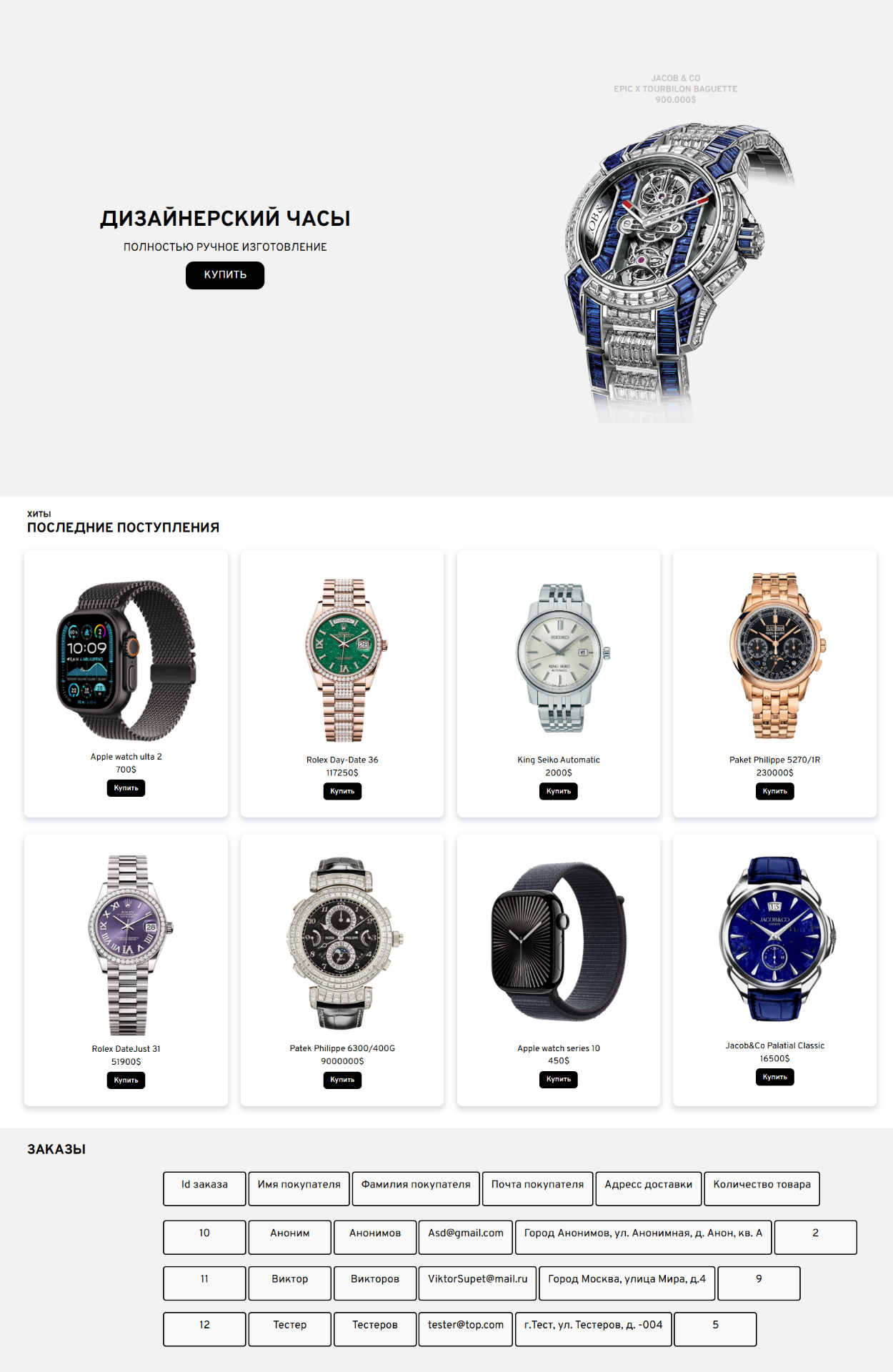


Рисунок 5. Модифицированное веб-приложение.

ВЫВОД К РАБОТЕ:

Я познакомился с технологией создания и использования веб-сервисов.