1. **ОБЩИЕ СВЕДЕНИЯ**

Полное наименование системы и её обозначение:

Наименование программы: Сайт интернет-магазина цветов.

Шифр темы: ПМ.03 Сопровождение и обслуживание программного обеспечения компьютерных систем

Наименование предприятия, разработчика и заказчика:

Задание на разработку программы было выдано в Энгельсском технологическом институте (филиал) Саратовского Государственного Технического Университета. Перечень документов, на основании которых создается система:

* Техническое задание
* Руководство пользователя

1. **НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ (РАЗВИТИЯ) СИСТЕМЫ**

2.1 Назначение системы

Система интернет-магазина цветов предоставляет пользователям возможность просматривать каталог товаров, оформлять заказы, а также авторизоваться и регистрироваться для удобного управления своими заказами.

1. **ХАРАКТЕРИСТИКА ОБЪЕКТОВ АВТОМАТИЗАЦИИ**

Краткие сведения об объекте автоматизации или ссылка на документы, содержащие такую информацию. Документы, на основании которых ведётся разработка:

* Техническое задание
* Руководство пользователя

1. **ТРЕБОВАНИЯ К СИСТЕМЕ**

4.1 Требования к системе в целом

4.1.1. Требования к структуре и функционалу системы Программа должна обеспечивать выполнение следующих функций:

* Разработка главной страницы с отображением популярных букетов;
* Отображение акционных предложений;
* Каталог цветов и цветочных композиций;
* Создание базы данных для хранения информации о товарах, заказах и пользователях;
* Авторизация и регистрация клиентов.

4.1.2. Требования к численности и квалификации персонала Для нормального функционирования системы требуется минимум 2 сотрудника, обладающих следующими навыками:

* Уверенное владение ПК;
* Базовые знания в области работы с удаленными базами данных (SQL).

4.1.3. Показатели назначения Целевые показатели внедрения системы:

* Отображение популярных букетов на главной странице;
* Представление акционных предложений;
* Каталог цветов и цветочных композиций с удобной навигацией;
* Создание и управление базой данных товаров, заказов и пользователей;
* Удобная авторизация и регистрация клиентов. Результатом внедрения программы является выполнение всех указанных функций, обеспечивающих удобство работы пользователей и администрации.

4.1.4. Требования к надежности Надежная работа системы обеспечивается соответствием платформы следующим минимальным требованиям:

* Процессор с тактовой частотой от 1.66 ГГц;
* Оперативная память не менее 2 ГБ;
* Видеоускоритель с поддержкой DirectX 9;
* Операционная система Windows 10;
* Разрешение экрана от 800x600;
* Наличие стандартных устройств ввода (клавиатура, мышь).

4.1.5. Требования к безопасности

* Предотвращение аппаратных сбоев;
* Защита данных от несанкционированного доступа и утечек.

4.1.6. Требования к транспортабельности Для данной системы требования к транспортабельности отсутствуют.

4.1.7. Требования к эксплуатации, обслуживанию и хранению Эксплуатация системы должна соответствовать требованиям СанПиН 2.2.2/2.4.1340-03:

* Организация рабочего места с учетом гигиенических норм;
* Регулярное техническое обслуживание серверного оборудования и баз данных.

4.1.8. Требования к защите от влияния внешней среды

* При перегрузке системы должно быть предусмотрено автоматическое отключение для предотвращения повреждения данных и оборудования;
* Соответствие аппаратных средств допустимым уровням влажности и температур, установленным производителем.

4.2. Требования к функциям, выполняемым системой

1. Главная страница — отображение популярных букетов и акционных предложений.
2. Каталог товаров — просмотр цветов и цветочных композиций.
3. База данных — создание и управление товарами, заказами и пользователями.
4. Авторизация и регистрация — возможность входа в систему и создания учетной записи.

4.3. Требования к видам обеспечения

Таблица 1. Функции системы

|  |  |  |
| --- | --- | --- |
| **Название операции** | **Функции** | **Действие** |
| Отображение популярных букетов | Вывод популярных товаров на главную страницу | Пользователь видит наиболее популярные цветочные композиции на главной странице. |
| Отображение акционных предложений | Вывод текущих скидок и акций | Пользователь получает информацию о действующих акциях и специальных предложениях. |
| Каталог цветов и композиций | Просмотр доступных товаров | Пользователь может просматривать ассортимент цветов и цветочных композиций. |

**СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ СИСТЕМЫ**

Перечень стадий и этапов работы по созданию системы, сроки выполнения:

Таблица 2. Этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| **Название этапа** | **Планируемая дата исполнения** | **Фактическая дата исполнения** | **Подпись исполнителя** |
| Ознакомление с основной нормативной документацией |  |  | **Михеева Э.А.** |
| Сформировать отчет об анализе предметной области |  |  | **Михеева Э.А.** |
| Заполнение информационных таблиц |  |  | **Михеева Э.А.** |
| Создание форм, оформление интерфейса |  |  | **Михеева Э.А.** |
| Написание кода |  |  | **Михеева Э.А.** |
| Тестирование работоспособности |  |  | **Михеева Э.А.** |
| Пробный запуск |  |  | **Михеева Э.А.** |

**АННОТАЦИЯ**

Настоящий документ представляет собой руководство пользователя (далее Руководство) интернет-магазина.

Сайт предоставляет пользователям заказывать цветочные композиции, оформлять доставку, настраивать персонализацию букетов, а также управлять заказами и клиентами через административную панель. Функции сайта интернет-магазина цветов

Руководство кратко описывает программу, а также определяет порядок установки на компьютер, дальнейшей эксплуатации и порядок действий при возникновении ошибок.

**УСЛОВИЯ ПРИМЕНЕНИЯ**

**Минимальные системные требования для клиентского приложения:**

* + Операционная система: Windows 10 или выше, macOS 10.15 или выше, современные дистрибутивы Linux.
  + Процессор: Двухъядерный процессор с тактовой частотой 1.8 ГГц или выше.
  + Оперативная память: 4 ГБ ОЗУ.
  + Хранилище: 500 МБ свободного места на диске для кеширования данных.
  + Экран: Разрешение не менее 1024x768.
  + Подключение к интернету: Стабильное соединение со скоростью 5 Мбит/с или выше.
  + Браузер: Современные версии Google Chrome, Mozilla Firefox, Microsoft Edge или Safari с поддержкой HTML5 и CSS3.

**Минимальные программные требования для клиентского приложения:**

* + Поддержка JavaScript и Cookies в браузере.
  + Наличие установленного браузера с актуальными обновлениями безопасности.
  + При использовании мобильных устройств – ОС Android 9.0 или выше, iOS 13.0 или выше.

**Минимальные системные требования для серверной части (рассчитано до 500 пользователей):**

* Операционная система: Windows Server 2019 или выше.
* Процессор: Четырехъядерный процессор с тактовой частотой 2.4 ГГц или выше.
* Оперативная память: 8 ГБ ОЗУ.
* Хранилище:
* 50 ГБ свободного места на диске (SSD рекомендуется) для базы данных и файлового хранилища.
* Сеть: Скорость соединения 10 Мбит/с или выше, с пропускной способностью, обеспечивающей обслуживание 500 одновременных пользователей.
* Программное обеспечение:
* Веб-сервер: Apache 2.4 или выше, либо Nginx 1.18 или выше.
* Интерпретатор: PHP 7.4 или выше с необходимыми модулями.
* СУБД: Po (в зависимости от выбранной архитектуры).
* Средства резервного копирования данных.

**КРАТКОЕ ОПИСАНИЕ ВОЗМОЖНОСТЕЙ**

Сайт интернет-магазина цветов предназначен для удобного и быстрого заказа цветочных композиций, оформления доставки, а также настройки персонализации букетов. Он упрощает процесс выбора и покупки цветов, обеспечивая удобное взаимодействие между покупателями и продавцами, а также предоставляет возможности для эффективного управления заказами и клиентами через административную панель.

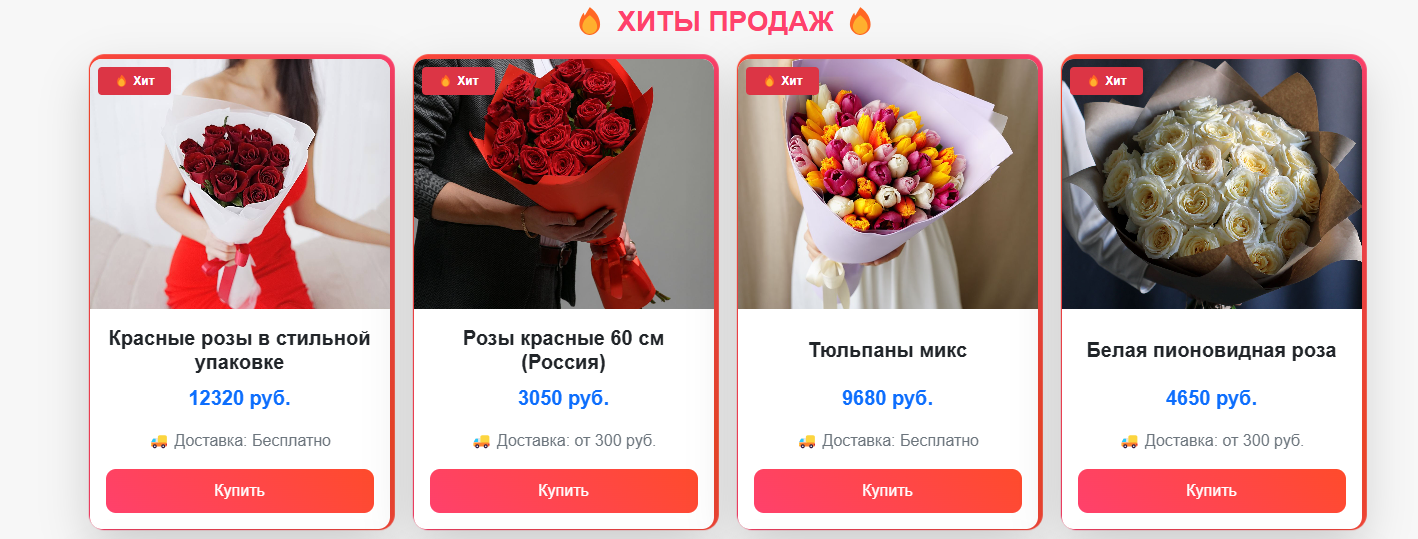
Функционал сайта позволяет пользователям легко находить нужные цветочные композиции, выбирать аксессуары и настроить индивидуальное оформление букетов. Благодаря интегрированным инструментам для выбора доставки, покупатели могут выбрать удобное время и место получения заказанных товаров. Также на сайте предусмотрены возможности для отслеживания статуса заказов и получения уведомлений.

Для администраторов сайта предусмотрены мощные инструменты для управления каталогом продукции, обработки заказов, взаимодействия с клиентами и настройки системы доставки. Модерация комментариев и отзывов о товарах, контроль за исполнением заказов и своевременное обновление информации — все это способствует улучшению качества обслуживания и повышению удовлетворенности клиентов.

Таким образом, сайт интернет-магазина цветов помогает создать удобную платформу для покупок, поддерживает персонализированный подход к каждому клиенту и обеспечивает высокое качество обслуживания на всех этапах заказа.

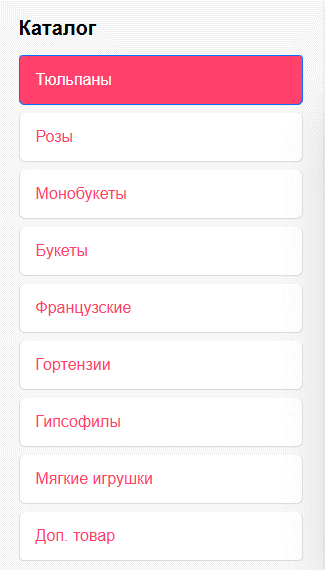
**ЭКСПЛУАТАЦИЯ**

На главной странице отображены основной блок с хитами продаж, а также с акциями (Рисунок 1.)



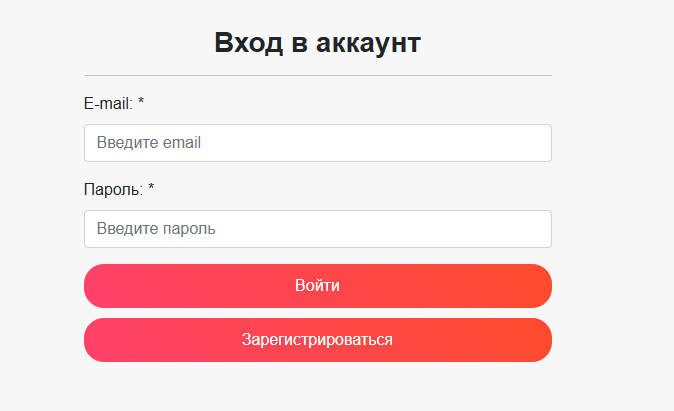
**Рисунок 1. Главная с хитами и акциями**

В каталоге, пользователь, имеет возможность выбора одного из представленных в категориях элемент (Рисунок 2).

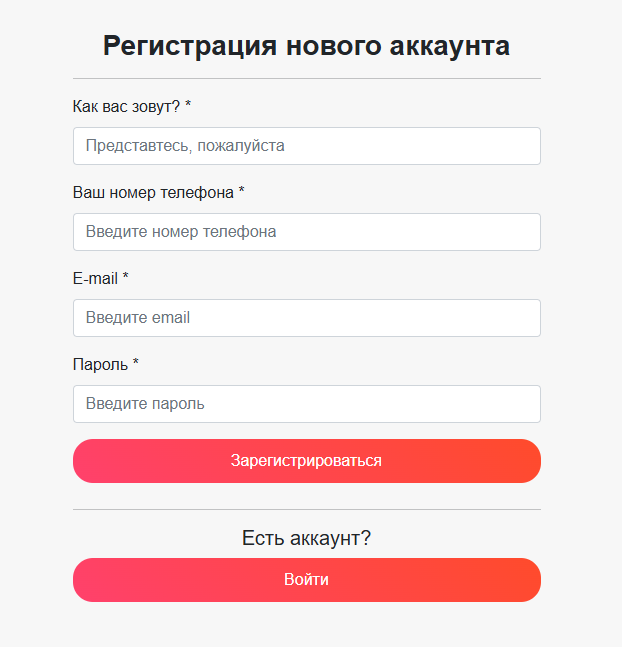


**Рисунок 2. Каталог**

Пользователь может зарегистрировать (Рисунок 4), или войти (Рисунок 3) в уже созданный аккаунт по основным данным.



**Рисунок 3. Авторизация**



**Рисунок 4. Регистрация**

**УСТАНОВКА ПРОГРАММЫ**

1. Нажать два раза правой кнопкой мыши по браузеру.
2. Заходим на сайт <http://petalswind.somee.com>
3. Установка завершена.

**ОШИБКИ И ПУТИ УСТРАНЕНИЯ**

Таблица 1. Ошибки и пути устранения

|  |  |  |  |
| --- | --- | --- | --- |
| Номер ошибки | Обозначение | Описание | Пути решения |
| 0001 | Не верный логин или пароль | Введены неверные логин или пароль | 1) Введите другие логин или пароль  2) Обратитесь к администратору для восстановления |
| 0002 | Не заполнены обязательные поля | Нужные поля, не заполнены. | Заполните все пустующие поля |

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Mvc;

using Лепестки\_ветра.Service;

namespace Лепестки\_ветра.Controllers

{

public class AccountController : Controller

{

private readonly AccountService \_authService;

// Единый конструктор

public AccountController(AccountService authService)

{

\_authService = authService ?? throw new ArgumentNullException(nameof(authService));

}

public IActionResult login()

{

return View();

}

[HttpPost]

public async Task<IActionResult> login(string email, string password)

{

if (string.IsNullOrWhiteSpace(email) || string.IsNullOrWhiteSpace(password))

{

ModelState.AddModelError("", "Email и пароль обязательны.");

return View();

}

try

{

var user = await \_authService.LoginAsync(email, password);

if (user == null)

{

ModelState.AddModelError("", "Неверный email или пароль.");

return View();

}

// Сохранение данных пользователя в сессию

HttpContext.Session.SetInt32("ID", user.id);

HttpContext.Session.SetString("Email", user.email);

HttpContext.Session.SetString("YourName", user.your\_name);

// Редирект на главную страницу

return RedirectToAction("Index", "Home");

}

catch (Exception ex)

{

// Логирование ошибки

ModelState.AddModelError("", "Произошла ошибка. Попробуйте позже.");

return View();

}

}

public IActionResult register()

{

return View();

}

[HttpPost]

public async Task<IActionResult> register(string email, string password,string your\_name, string telephone)

{

var user = await \_authService.RegisterAsync(email, password, your\_name, telephone);

if (user == null)

{

ModelState.AddModelError("", "Пользователь с таким email уже существует.");

return View();

}

return RedirectToAction("Login");

}

// Метод для выхода из аккаунта

[HttpPost]

public async Task<IActionResult> Logout()

{

await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);

HttpContext.Session.Clear();

return RedirectToAction("Index", "Home");

}

public IActionResult profile()

{

return View();

}

}

}

using Microsoft.AspNetCore.Mvc;

public class CatalogController : Controller

{

private readonly ProductService \_productService;

public CatalogController(ProductService productService)

{

\_productService = productService;

}

public async Task<IActionResult> catalog(string category = "Тюльпаны")

{

var products = await \_productService.GetProductsByCategoryAsync(category);

var viewModel = products.Select(p => new ProductViewModel

{

Id = p.id,

Name = p.name,

Price = p.price,

ImageUrl = p.imageurl,

IsHit = p.is\_hit,

DeliveryInfo = p.delivery

}).ToList();

ViewBag.CurrentCategory = category; // Передаем текущую категорию в представление

return View(viewModel);

}

}

using System.Diagnostics;

using Лепестки\_ветра.Models;

using Microsoft.AspNetCore.Mvc;

namespace Лепестки\_ветра.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

private readonly ProductService \_productService;

public HomeController(ILogger<HomeController> logger, ProductService productService)

{

\_logger = logger;

\_productService = productService;

}

public async Task<IActionResult> Index()

{

var product = await \_productService.GetProductAsync();

return View(product);

}

public async Task<IActionResult> Catalog()

{

var product = await \_productService.GetProductAsync();

return View(product);

}

public IActionResult Privacy()

{

return View();

}

public IActionResult About()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

using Supabase.Gotrue;

using Лепестки\_ветра.Models;

namespace Лепестки\_ветра.Service

{

public class AccountService

{

private readonly Supabase.Client \_client;

public AccountService(SupabaseClientService clientService)

{

\_client = clientService.Client;

}

public async Task<User?> LoginAsync(string email, string password)

{

try

{

var user = await \_client.From<User>()

.Filter("email", Supabase.Postgrest.Constants.Operator.Equals, email)

.Single();

if (user == null)

{

return null;

}

var passwordHasher = new Microsoft.AspNetCore.Identity.PasswordHasher<User>();

var result = passwordHasher.VerifyHashedPassword(user, user.password, password);

if (result != Microsoft.AspNetCore.Identity.PasswordVerificationResult.Success)

{

return null;

}

return user;

}

catch (Exception ex)

{

Console.WriteLine($"Error in LoginAsync: {ex.Message}");

return null;

}

}

// Регистрация

public async Task<User?> RegisterAsync(string email, string password, string your\_name, string telephone)

{

try

{

var existingUser = await \_client.From<User>()

.Filter("email", Supabase.Postgrest.Constants.Operator.Equals, email)

.Single();

if (existingUser != null)

{

return null;

}

var passwordHasher = new Microsoft.AspNetCore.Identity.PasswordHasher<User>();

var hashedPassword = passwordHasher.HashPassword(null, password);

var newUser = new User

{

email = email,

password = hashedPassword,

your\_name = your\_name,

telephone = telephone

};

var response = await \_client.From<User>().Insert(newUser);

return response.Models.FirstOrDefault();

}

catch (Exception ex)

{

return null;

}

}

}

}

using Supabase;

using Supabase.Gotrue;

using Supabase.Postgrest;

using Supabase.Postgrest.Exceptions;

using Лепестки\_ветра.Models;

public class ProductService

{

private readonly Supabase.Client \_client;

public ProductService(SupabaseClientService clientService)

{

\_client = clientService.Client;

}

public async Task<List<Product?>> GetProductAsync()

{

var response = await \_client.From<Product>().Get();

return response.Models;

}

public async Task<List<Product>> GetProductsByCategoryAsync(string category)

{

var response = await \_client.From<Product>()

.Where(p => p.type == category)

.Get();

return response.Models;

}

}

public class SupabaseClientService

{

private readonly Supabase.Client \_client;

public SupabaseClientService(string supabaseUrl, string supabaseKey)

{

\_client = new Supabase.Client(supabaseUrl, supabaseKey);

}

public async Task InitializeAsync()

{

await \_client.InitializeAsync();

}

public Supabase.Client Client => \_client;

}