**Введение**

Метеорология – наука об атмосфере, ее составе, строении, свойствах, физических и химических процессах, в ней происходящих. Теоретической основой метеорологии служат фундаментальные законы физики и химии.

Температура воздуха – один из термодинамических параметров состояния атмосферы.

Прогноз погоды – научно обоснованное предположение о будущем состоянии погоды в определённом пункте или регионе на определённый период.

1. **Постановка задачи**

Разработка проводятся на основании Договора Между ЛДПК и ООО «Evion» № 17 от 17 апреля 2023 г.

ООО «Evion»

Наименование работы: Настольно приложение “WeatherApp”.

Программа предназначена для просмотра данных о погоде пользователем.

Каждый пользователь проходит обязательную регистрацию, при которой данные его учетной записи сохраняются в базу данных.

Приложение позволяет узнать минимальную температуру за период, максимальную температуру за период, среднюю температуру за период, текущую погоду, посмотреть график изменения погоды.

**2. Проектирование интерфейса приложения**

**2.1. Правила и принципы разработки интерфейса (теория)**

**Интерфейс** – внешняя оболочка приложения, дающая возможность работы с любым типом информации, хранящейся на компьютере (для этого имеется внутренние и внешние механизмы управления информацией). процесс проектирования интерфейса – циклический процесс, обладающий двумя признаками:

1. Проектирование каскадной схемы;
2. Проектирование спиральной схемы.

**Типы проектирования.**

**Стандартизация** – рекомендуется использовать стандартные, проверенные многими программистами и пользователями интерфейсные решения.

**Способы выделения интерфейса яркостью:**

1. Движение (мигание, изменение позиции). Относится к эффективным методам, поскольку человеческий глаз восприимчив к движению.
2. Изменение яркости 0% - 100%.
3. Окружение – подчеркивание либо инвертированное изображение.
4. Оттенение – разная текстура объектов.
5. Средства управления графическим интерфейсом пользователя.

**Элементы управления:**

1. Кнопки
2. CheckBox
3. Списки (ListBox, ComboBox)
4. Текстовые поля и метки (TextBox, Label)
5. Изображения

**Типы интерфейсов.**

1. Однодокументный SDI:
2. главное меню
3. панели инструментов с элементами управления (задается пользователем)
4. окно приложения для размещения элементов управления (рабочее поле)
5. строка состояния
6. Многодокументный MDI:
7. главное меню
8. панель инструментов с элементами управления
9. главное окно
10. Проводник Explorer.

**2.2. Разработка макета приложения**

**3. Структура хранения данных**

Данные хранятся в базе данных PostgreSQL. База данных создается при запуске веб-сервера.

**3.1. Проектирование структуры хранения данных (БД)**

Базы данных (БД) – хранилище структурированных данных и методы доступа к ним, при этом данные должны быть непротиворечивы, целостны и минимально избыточны. Это особым образом организованный набор значений данных, а схема БД определяет, как именно организованы данные в БД. Можно сказать, что БД создаются для хранения и доступа к данным, содержащим сведения о некоторой предметной области, т.е. всякая БД представляет собой систему данных о предметной области.

На этом этапе анализируется имеющаяся информация с целью определить набор таблиц и их полей. Проектирование базы данных заключается в определении состава полей ее таблицы и связей между таблицами. От того, насколько тщательно проведен анализ и насколько грамотно спроектирована БД, в существенной мере зависит эффективность будущей программы и ее полезность для пользователя. Для того, чтобы создать БД, ее нужно нормализовать.

Предметная область, которая исследуется это информация о городах и температуре воздуха в них. В результате выявлены следующие сущности: координаты города (широта и долгота), наименование городов и температура воздуха в городах за периоды времени.

В процессе разработки структуры базы данных может возникнуть избыточность информации – это повторение (дублирование) данных, содержащихся в базе данных. В таблице с избыточными данными содержится много повторяющихся данных в различных записях. Для избавления от избыточности можно разбить одну большую таблицу на несколько маленьких. Такой процесс разбиения таблицы называется нормализацией базы данных. Т.е. нормализация – это процесс устранения избыточности данных.

Существует несколько так называемых нормальных форм (НФ) баз данных. Каждая из нормальных форм получается последовательно из предыдущей нормальной формы. На практике применяются в основном первые три из них.

Первая нормальная форма требует, чтобы каждое поле таблицы БД было не делимым, не содержало повторяющихся групп. Не делимость поля означает, что каждое поле не должно делится на несколько полей. А повторяющиеся группы указывают на то, чтобы поля не содержали одинаковые по смыслу значения.

Вторая нормальная форма, требует: все поля таблицы зависели от первичного ключа, то есть первичный ключ однозначно определен и является не избыточным; те поля, которые зависят от части первичного ключа, должны быть выделены в отдельные таблицы.

Третья нормальная форма требует, чтобы значение любого поля таблицы, не входящего в первичный ключ, не зависело от значения другого поля, не входящего в первичный ключ.

Нормализованной БД называется база, в которой выполняется как минимум три условия. В результате нормализации получаться следующие таблицы:

Таблица «cities» содержит данные о городах: наименование, координаты.

Таблица №1 «cities»

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | city | longtitude | latitude |
| 15 | Ликино-Дулево | 38.9542 | 55.7083 |
| 16 | Орехово-Зуево | 38.96178 | 55.80672 |

Таблица «city\_info» содержит данные о погоде для каждого города: максимальная, минимальная, средняя и текущая температура воздуха.

Таблица №2 «city\_info»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | min\_weather | max\_weather | period\_weather | current\_weather | city\_id |
| 15 | Ликино-Дулево | 5,4,5,1,4,5,1 | 3,4,3,5,3,1 | 4,5,6,7,8,9 | 15 |
| 16 | Орехово-Зуево | 6,5,2,5,6,2 | 3,4,3,3,5,1 | 4,5,7,7,8,9 | 16 |

Таблица «users» содержит данные о пользователях, их логины и пароли.

Таблица №3 «users»

|  |  |  |
| --- | --- | --- |
| **id** | username | password |
| 1 | admin | admin |
| 2 | user | user |

Таблица «user\_info» содержит данные городах, который добавил для себя каждый юзер.

Таблица №4 «user\_info»

|  |  |  |
| --- | --- | --- |
| **id** | user\_id | city\_id |
| 1 | 1 | 15 |
| 2 | 2 | 16 |

**3.2. Наполнение структуры данными (вручную, импорт…, где берется информация…)**

Данные заполняются автоматически, получая данные по API.

**4. Разработка технической документации**

**4.2. Руководство программиста**

**4.2.1. Выбор средств разработки**

C# (Си-шарп) — объектно-ориентированный язык программирования для платформы .NET. Разработан в 2000 году Андерсом Хейлсбергом, Скоттом Вилтамутом и Питером Гольде под эгидой Microsoft Research. Основным постулатом С# является высказывание: "всякая сущность есть объект". Язык основан на строгой компонентной архитектуре и реализует передовые механизмы обеспечения безопасности кода.

C# был создан специально для технологии ASP.NET. В то же время на C# полностью написана и сама ASP.NET.

C# — это полнофункциональный объектно-ориентированный язык, который поддерживает все три «столпа» объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. Он имеет прекрасную поддержку компонентов, надежен и устойчив благодаря использованию «сборки мусора», обработки исключений, безопасности типов.

Язык C# разрабатывался "с нуля" и вобрал в себя много полезных свойств таких языков, как C++, Java, Visual Basic, а также Pascal, Delphy и др. При этом необходимость обратной совместимости с предыдущими версиями отсутствовала, что позволило языку C# избежать многих отрицательных сторон своих предшественников.

Как и Java, C# разрабатывался для Интернет и примерно 75% его синтаксических возможностей аналогичны языку программирования Java, его также называют «очищенной версией Java. 10% подобны языку программирования C++, а 5% – заимствованы из языка программирования Visual Basic. Объем новых концептуальных идей в языке C# около 10%.

Выделение и объединение лучших идей современных языков программирования делает язык C# не просто суммой их достоинств, а языком программирования нового поколения.

**4.2.2. Технологии доступа к данным**

Технологии доступа к данным являются прослойкой между API конкретного сервера и приложением пользователя, предоставляя программисту простой унифицированный механизм работы с данными.

На сегодняшний день существует множество технологий доступа к данным, таких как BDE, OLE, ODBC, DАО, АDО, и до сих пор разрабатываются новые, более надежные, удобные в работе и более быстродействующие технологии.

Механизмы доступа к базам данных снижают сложность обмена информацией с базами, однако интерпретация результатов их работы также достаточно трудоемка. Поэтому реализованы наборы компонентов, предназначенные для взаимодействия с механизмами обмена.

Таким образом, можно выделить несколько субъектов, участвующих в движении информации между базой данных и приложением (например, пользовательским интерфейсом):

1) интерфейсная часть приложения или его программная часть, манипулирующая информацией, хранимой в базе данных.;

2) компоненты, обеспечивающие связь приложения с механизмом доступа к базе данных;

3) механизм доступа к базе данных;

4) база данных.

**4.2.3. Программные методы разработки интерфейса**

Пользовательский интерфейс представляет собой совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи: обмен информацией и координация действий. Каждый диалог состоит из отдельных процессов ввода-вывода, которые физически обеспечивают связь пользователя и компьютера.

Обмен информацией осуществляется передачей сообщений и управляющих сигналов. Сообщение - порция информации, участвующая в диалоговом обмене. Различают:

• входные сообщения, которые генерируются человеком с помощью средств ввода: клавиатуры, манипуляторов, например мыши и т. п.

• выходные сообщения, которые генерируются компьютером в виде текстов, звуковых сигналов и/или изображений и выводятся пользователю на экран монитора или другие устройства вывода информации.

В основном пользователь генерирует сообщения следующих типов: запрос информации, запрос помощи, запрос операции или функции, ввод или изменение информации, выбор поля кадра и т. д. В ответ он получает: подсказки или справки, информационные сообщения, не требующие ответа, приказы, требующие действий, сообщения об ошибках, нуждающиеся в ответных действиях, изменение формата кадра и т. д.

Ниже перечислены основные устройства, обеспечивающие выполнение операций ввода-вывода.

Для вывода сообщений:

• монохромные и цветные мониторы - вывод оперативной текстовой и графической информации;

• принтеры - получение «твердой копии» текстовой и графической информации;

• графопостроители - получение твердой копии графической информации;

• синтезаторы речи - речевой вывод;

• звукогенераторы - вывод музыки и т. п.

Для ввода сообщений:

• клавиатура - текстовый ввод;

• планшеты - графический ввод;

• сканеры - графический ввод;

• манипуляторы, световое перо, сенсорный экран - позиционирование и выбор информации на экране и т. п.

**4.2.4. Реализация функционала приложения (основные на выбор процедуры, функции…)**

Функции серверной части:

Метод GetGEO(string city) получает координаты города, который добавляет пользователь:

static async Task<string> GetGEO(string city)

{

HttpClient httpClient = new HttpClient();

// получаем ответ

using HttpResponseMessage response = await httpClient.GetAsync("https://geocoding-api.open-meteo.com/v1/search?name=" + city + "&language=ru");

// получаем ответ

return await response.Content.ReadAsStringAsync();

}

Метод get\_current(string city\_id) возвращает данные о выбранном городе:

public double get\_current(string city\_id)

{

var res = WeatherContext.GetContext().CityInfos.Where(x => x.CityId == int.Parse(city\_id)).ToList();

return res[0].CurrentWeather;

}

Метод Login(string username, string password) проверяет введеные пользователем логин и пароль, и в случае соответствия дает пользователю доступ к программе:

public string Login(string username, string password)

{

var res = WeatherContext.GetContext().Users.Where(x => x.Password == password && x.Username == username).ToList();

if (res.Count == 1) return "Success";

else return "Failed";

}

Функции приложения:

Метод BtnAuth\_Click(object sender, RoutedEventArgs e) производит отправку логина и пароля на сервер для авторизации пользователя:

private async void BtnAuth\_Click(object sender, RoutedEventArgs e)

{

string username = TxbLogin.Text;

string password = PsbPassword.Password.ToString();

var response = await Auth(username, password);

if (response.ToString() == "Success")

{

User.Password = password;

User.Username = username;

manager.MainFrame.Navigate(new Weath());

}

else { MessageBox.Show("Неверные данные!"); };

}

static async Task<string> Auth(string username, string password)

{

HttpClient httpClient = new HttpClient();

var dt = DateTime.Now;

DateTime month = dt.AddMonths(-1);

// получаем ответ

using HttpResponseMessage response = await httpClient.GetAsync("https://localhost:7205/login/"+username+"/"+password);

return await response.Content.ReadAsStringAsync();

}

Аналогичный метод для регистрации:

private async void BtnAuth\_Click(object sender, RoutedEventArgs e)

{

string username = TxbLogin.Text;

string password = PsbPassword.Password.ToString();

var response = await Auth(username, password);

if (response.ToString() == "Success")

{

User.Password = password;

User.Username = username;

manager.MainFrame.Navigate(new Weath());

}

else { MessageBox.Show("Данный логин уже занят!"); };

}

static async Task<string> Auth(string username, string password)

{

HttpClient httpClient = new HttpClient();

var dt = DateTime.Now;

DateTime month = dt.AddMonths(-1);

// получаем ответ

using HttpResponseMessage response = await httpClient.GetAsync("https://localhost:7205/register/" + username + "/" + password);

return await response.Content.ReadAsStringAsync();

}

**4.2.5. Обработки исключений**

Участок кода с получением данных о температуре за неделю, работающий некорректно:

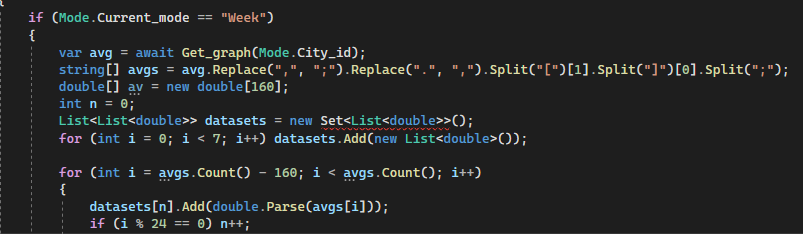


Рис.1 «Неявное присваивание»

Исправленный участок кода с получением данных о температуре за неделю, работающий корректно:

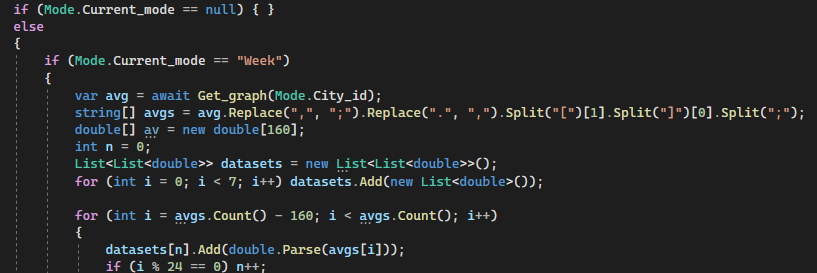


Рис.2 «Присваивание списка списков чисел»

Участок кода с переходом на страницу добавления города, работающий некорректно:

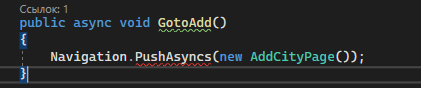


Рис.3 «Неправильно указано имя модуля»

Участок кода с переходом на страницу добавления города, работающий корректно:

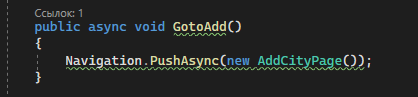


Рис.4 «Указания правильного имени модуля»

**4.3. Руководство пользователя (алгоритм работы с программой…)**

При открытии программы (исполняемый «.exe» файл) появится страница авторизации, где пользователь может войти в сиcтему используя логин и пароль, или перейти на страницу регистрации.

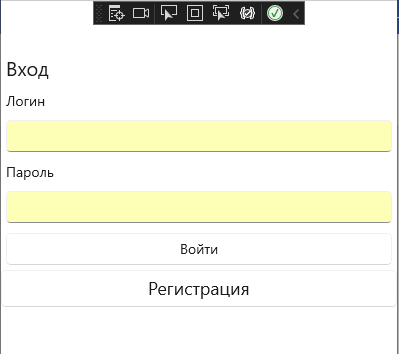


Рис. 5 «Страница авторизации»

После нажатия на кнопку «Войти» пользователь попадает в главное меню программы. На данной страницы с помощью элементов управления можно выбрать город, добавить или удалить его, выбрать период, за который отображать данные о температуре.

При нажатии на кнопку «График» откроется страница с графиком температуры.

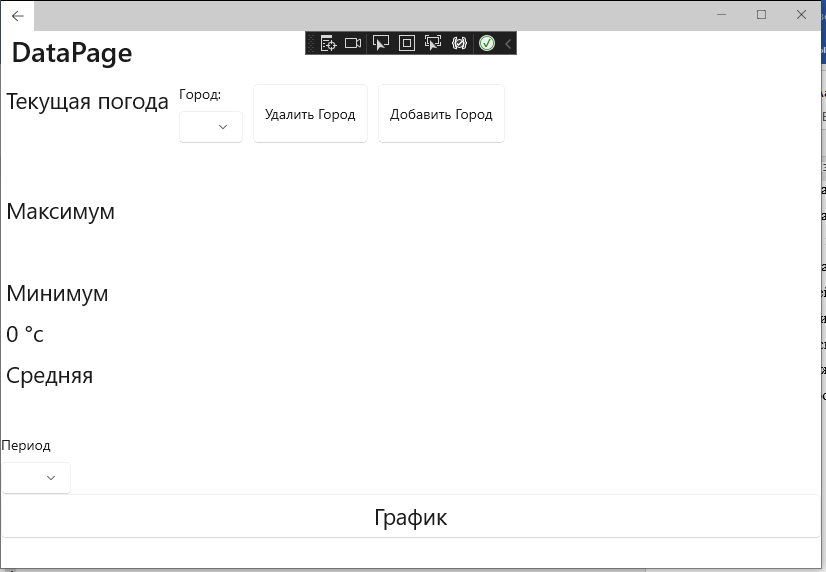


Рис. 6 «Главная страница»

При раскрытии списка «Город» пользователю необходимо выбрать город, данные о котором он хочет увидеть.

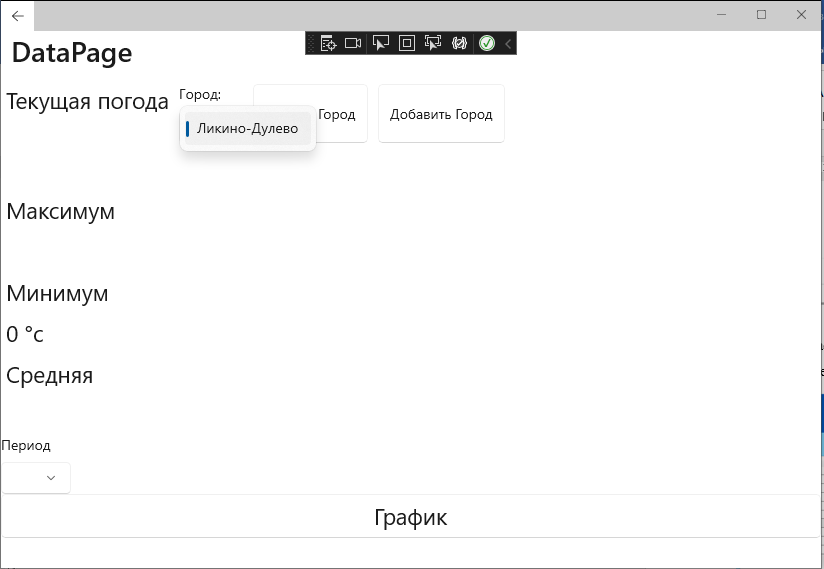


Рис. 7 «Список городов»

При раскрытии списка «Период» пользователю необходимо выбрать период, в котором он хочет увидеть данные о температуре в выбранном городе.

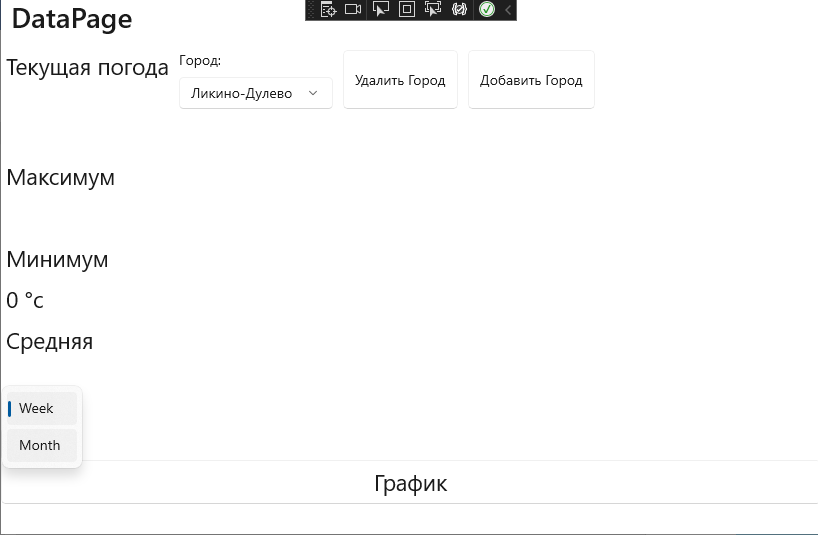


Рис. 8 «Список периодов»

На странице «Добавление города» необходимо ввести название города и нажать кнопку «Добавить».

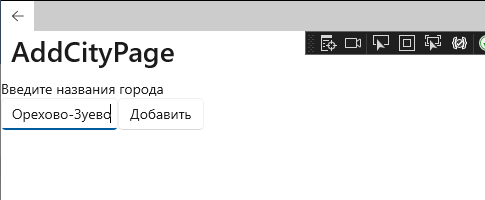


Рис. 9 «Страница добавления города»

На странице «График» отображается график изменения температуры в выбранном городе за выбранный период.

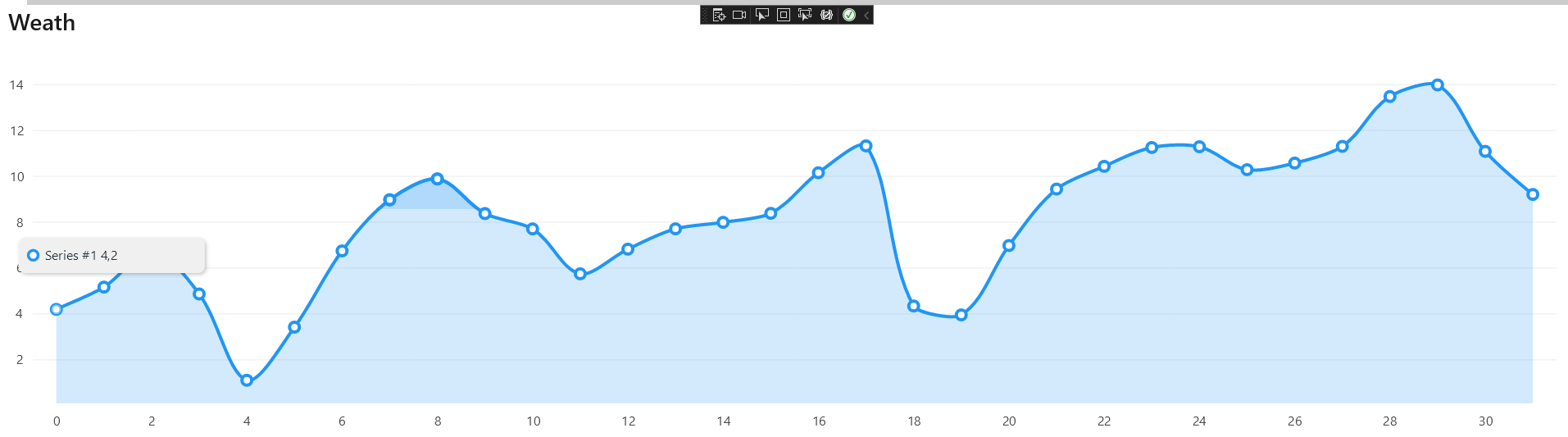


Рис. 10 «Страница «График»»

**Заключение**

В заключение хочу сказать, что представленный сайт способен успешно осуществлять выполнение поставленных перед ним задач, однако его функциональные возможности могут быть расширены. В будущем на сайт можно было бы добавить функцию корзины. Необходимо расширить количество данных в базе данных. В процессе разработки данного сайта, я углубил свои знания и навыки в области информационных технологий, изучив множество новых и полезных методик и технологий, которые я успешно применил в ходе работы. В процессе разработки были использованы такие технологии как:

PHP (Hypertext Preprocessor) - это скриптовый язык программирования, который широко используется для создания динамических веб-сайтов и веб-приложений. PHP выполняется на стороне сервера и может генерировать HTML, CSS, JavaScript и другие форматы контента для веб-браузера.

1С-Битрикс - это платформа для создания веб-сайтов и интернет-магазинов, основанная на языке программирования PHP и использующая базу данных MySQL. Она была разработана компанией 1С-Битрикс и предоставляет широкий набор инструментов для создания и управления веб-проектами любого уровня сложности.

**Список литературы**

1. Басыров, Р. 1С-Битрикс. Корпоративный портал. Повышение эффективности компании (+ CD-ROM) / Р. Басыров. - М.: Питер, 2010. - 320 c.  
2. Басыров, Р. 1С-Битрикс. Строим профессиональный сайт и интернет-магазин / Р. Басыров. - М.: Книга по Требованию, 2011. - 544 c.  
3. Басыров, Роберт 1С-Битрикс: Корпоративный портал. Руководство разработчика/Роберт Басыров.-М.: Рид Групп, 2012. - 352 c.  
4. Рассел, Джесси 1С-Битрикс / Джесси Рассел. - М.: Книга по Требованию, 2012. - 546 c.  
5. Расторгуев, О. П. Сайт на 1С-Битрикс. Создание, поддержка и продвижение. Базовое практическое руководство / О.П. Расторгуев, Р.Г. Прокди. - М.: Наука и техника, 2012. - 256 c.

**Приложение**

**1. Техническое задание**

**2. Руководство программиста**

**3. Руководство пользователя**

**4. Тестирование программных модулей**