


30.09.2020 / 125/3-13

УТВЕРЖДАЮ

Директор департамента АСУТП

 К.С. Теличко

« 30 » июля 2021 г.


ВИЗУАЛИЗАЦИЯ РАЗГАРА ГОРНА ДП-4

Том 2. Организационное обеспечение

Книга 2. Руководство программиста

90.32.048.ИЗ-02-М

Гл. специалист группы ДПиЭ

 А.В. Суковицын

« 30 » июля 2021 г.

Математик группы ДПиЭ

 Г.Р. Долгий

« 30 » июля 2021 г.

АННОТАЦИЯ

Данная работа выполняется на основании проекта «ДП №4 №GS18.233R Gongyi».

В документе представлено описание назначение и условия применения приложения, характеристика программы, обращение к программе, входные и выходные данные, сообщения аппаратной части приложения «Furnace Heat».

					90.32.048.ИЗ-02-М	Лист
						2
Изм.	Лист	№ докум.	Подпись	Дата		

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ	5
Функции программы	5
Условия применения	5
Требования к железу	5
Требования к ПО	6
2. ХАРАКТЕРИСТИКА ПРОГРАММЫ	7
Временные характеристики и режим работы	7
Средства контроля правильности выполнения	7
3. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ	8
База данных	8
Необходимые данные для подключения к БД <i>Domna4</i> :	8
Строка подключения к базе данных:	8
Работа с таблицами базы данных:	8
<i>IHeatRepository</i>	8
<i>IHistoryRepository</i>	10
<i>ISensorErrorRepository</i>	10
Контроллер	12
Функции контроллера:	12
Методы контроллера:	12
Промежуточные службы (Middleware)	14
<i>LoggingService</i>	14
Основные расчетные функции приложения	14
<i>complementData(data)</i>	14
<i>devideData(data)</i>	14
<i>gorn(basicPoyasa, poyasa)</i>	15
<i>leshad(poyasa)</i>	15
<i>razgarCalcRoot(coefs)</i>	15
<i>smallFunctions</i>	15
Общие классы приложения	15
Class <i>IHandler</i>	15
class <i>DevideHandler</i>	16
class <i>ComplementHandler</i>	16
class <i>LeshadHandler</i>	16
class <i>GornHandler</i>	16
class <i>SensorError</i>	16
class <i>SensorErrorHandler</i>	16
Основные графические элементы приложения	16
<i>NavMenu</i>	16
<i>Layout</i>	17
<i>Footer</i>	17
Представление «Разгар горна»	17
Функции представления	17
Основные компоненты представления «Разгар горна»	17
Алгоритм работы представления «Разгар горна»	18

Представление «Тренды».....	19
<i>Функции представления</i>	19
<i>Основные компоненты представления «Тренды»</i>	19
<i>Алгоритм работы состояния rWall</i>	21
<i>Алгоритм работы состояния Temp</i>	21
Представление «Статистика»	22
<i>Функции представления</i>	22
<i>Основные компоненты представления</i>	22
<i>Алгоритм работы представления</i>	22
4. СООБЩЕНИЯ.....	23
Проверка работы программы.....	23
<i>Начальная страница</i>	23
<i>Страница температурных трендов</i>	23
<i>Запрос не существующей страницы</i>	24
<i>Проверка работы контроллера</i>	24
<i>Запрос не существующей даты</i>	25
<i>Запрос даты вне информационной доступности</i>	25
Перечень возможных неисправностей	26
<i>Не исчезает значок Загрузка и не появляется меню</i>	26
<i>Не удается получить доступ к сайту</i>	27

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ

Веб-приложение «Furnace Heat» реализует алгоритм «Расчет состояния разгара горна и лещади» представленный в **Т6.К1.Описание алгоритма**. Назначение программы – визуально отобразить состояние горизонтальных и вертикальных сечений доменной печи № 4, показаний температурных датчиков, трендов температур датчиков и остаточной толщины огнеупорной кладки, прогноза динамики изменения этой толщины для оценки, контроля состояния футеровки печи, предотвращения аварий и принятия оперативных мер по увеличению срока службы печи до следующего капитального ремонта.

Программа реализована для работы в качестве веб-службы, что дает возможность ее использования с любого компьютера в сети ПАО «ЧМК».

Функции программы

- ❖ Предоставить визуальное представление температурных изотерм (300, 500, 800, 1150 °С на горизонтальных сечениях ДП-4 по поясам согласно проекту «ДП №4 №GS18.233R Gongyi» (всего 7 поясов начиная с 6),
- ❖ Предоставить визуальное представление рассчитанной изотермы 1150 °С на вертикальных сечениях ДП-4 (всего 32 луча),
- ❖ Для текущей даты предоставить поминутное обновление изотерм согласно мгновенным оперативным данным из таблицы минутных значений датчиков,
- ❖ Предоставить информацию о температурах датчиков за выбранный временной интервал в виде трендов,
- ❖ Предоставить информацию об остаточной толщине футеровки в виде трендов,
- ❖ Предоставить отображение прогноза толщины стенки исходя из усреднения накопленных данных по толщине стенки за все время,
- ❖ Предоставить статистику посещения сайта по запросу.

Условия применения

Для развертывания приложения необходим компьютер с установленной ОС Windows7 (Server 2008) или выше. Так как приложение способно работать даже на обычной станции, установка на компьютер с серверной версией ОС не обязательна. Необходимым условием работы приложения является рабочее состояние IIS.

Требования к железу

- ❖ 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц или выше, 1 ГБ (для 32-разрядного процессора) или 2 ГБ (для 64-разрядного процессора) ОЗУ,

					90.32.048.ИЗ-02-М	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

- ❖ Жесткий диск с частотой вращения шпинделя 5400 об/мин или SSD,
- ❖ 500 МБ свободного места на жестком диске,
- ❖ Видеоадаптер, совместимый с DirectX 9 или выше поддерживающий разрешение экрана 1440x900 (для контроля работы)

Требования к ПО

- ❖ В ОС компьютера должны быть подняты (доустановлены если это не серверная версия ОС) IIS (Internet Information Services).
- ❖ Приложение написано под ОС Windows и корректно функционирует в браузерах Chrome, Firefox и MS Edge.
- ❖ Приложение не предназначено для запуска посредством браузера MS Internet Explorer.

					90.32.048.ИЗ-02-М	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

2. ХАРАКТЕРИСТИКА ПРОГРАММЫ

Программа представляет собой REST-application. В качестве источника температур датчиков служит база данных расположенная на сервере 2 уровня доменной печи №4. Для осуществления запросов к базе и предоставлении информации в структурируемой форме служит HeatController написанный на C# asp .Net Core 3.1. Для отображения данных в веб браузере служит фреймворк React.

Временные характеристики и режим работы

Приложение реализовано в виде сайта на сервере служб ИС. Доступ к приложению возможен непрерывно в течение любого времени при условии работоспособности сервера. Особенность отображаемых данных характеризуется спецификой работы базы данных на сервере ГЦ. Так как за прошлые даты информация берется как среднее 30-ти минутных значений, отображение информации постоянно. При отображении текущей даты при выбранной опции авто обновления графики горизонтальных и вертикальных сечений могут изменяться с течением времени под воздействием изменяющихся данных.

Средства контроля правильности выполнения

Так как приложения является веб-службой, то контроль правильности выполнения возложен на ИС и веб-протоколы HTTP. В программе реализована защита от ввода некорректных данных (запрос дат вне предела информационной доступности, а также неверная последовательность ввода дат необходимого интервала). Подробное описание см. в главе СООБЩЕНИЯ.

					90.32.048.ИЗ-02-М	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

3. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

База данных

Источником информации о температурах датчиков ДП-4 служит БД Domna4. Запросы осуществляются к нескольким таблицам в базе данных.

Необходимые данные для подключения к БД Domna4:

- ❖ ip: 10.2.54.192,
- ❖ database: Domna4,
- ❖ authenticationType: SqlLogin,
- ❖ user: asu_user,
- ❖ password: asu_user,

Строка подключения к базе данных:

```
"server=tcp:10.2.54.192;  
Initial Catalog=Domna4;  
User ID=asu_user;  
Password=asu_user;"
```

Работа с таблицами базы данных:

Непосредственные обработчики запросов к базе данных реализованы с помощью паттерна «Репозиторий», содержащего основные CRUD операции и представляющие возможность расширять необходимый функционал запросов.

Интерфейсы репозитория реализованы посредством шаблонного интерфейса IRepository<T>, которому наследуют четыре дочерних IHeatRepository (для работы с температурами датчиков), IHistoryRepository (для работы с таблицей для хранения рассчитанных значений температуры 1150 °С, ISensorErrorRepository (для работы с ошибками датчиков печи) и IUsageLogRepository (для работы с таблицей статистики посещений сайта).

IHeatRepository

Предназначен для работы с температурами датчиков. Наследует интерфейсу IRepository<Sensor> и расширяет его тремя методами.

```
async Task<Dictionary<string, List<Sensor>>> Instant()
```

Этот метод служит для получения мгновенных данных из таблицы dbo.Minuta. SQL запрос:

```
SELECT  
    o.TagComment [name],  
    m.VarValue val,  
    sh.VarValue prevVal  
FROM ShortHistory sh
```

					90.32.048.ИЗ-02-М	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		


```

JOIN Settings_OPCTagsList o
  ON o.TagName LIKE '%T_Futerovki%'
  AND o.Tag_ID=sh.TagId
  AND sh.HH=DATEPART(HOUR, GETDATE())-1
  AND DATEPART(MINUTE, sh.[DateTime])=0
  AND FORMAT(sh.[DateTime], 'yyyy-MM-dd') = FORMAT(GETDATE(),
'yyyy-MM-dd')
JOIN Minuta m
  ON m.VarName LIKE '%T_Futerovki%'
  AND m.VarName=o.TagName
ORDER BY o.TagComment";

```

async Task<IEnumerable<Sensor>> GetFor(string dt)

Служит для получения показаний температурных датчиков за определенную дату. SQL запрос:

```

select
  o.TagComment name,
  avg(h.VarValue) val
from {0} h
join Settings_OPCTagsList o
  on h.TagId=o.Tag_ID
  and o.TagName like '%T_Futerovki%'
  and cast(h.[DateTime] as date) = cast('{1}' as date)
group by o.TagComment
order by o.TagComment

```

Вместо {0} подставляется таблица (для текущей даты – ShortHistory, для предыдущих дат – LongHistory). Вместо {1} подставляется необходимая дата в формате ГГГГ-ММ-ДД.

async Task<Dictionary<string, List<Sensor>>> GetFor(string bDate, string eDate)

Предоставляет структуру показаний датчиков в пределах запрошенного интервала дат. SQL запрос:

```

WITH interval AS (
  SELECT
    o.TagComment [name],
    h.VarValue val,
    FORMAT(h.[DateTime], 'yyyy-MM-dd') [date]
  FROM {0} h
  JOIN Settings_OPCTagsList o
    ON h.TagId=o.Tag_ID
    AND o.TagName LIKE '%T_Futerovki%'
    AND h.[DateTime] BETWEEN @bDate AND @eDate
)
SELECT
  [name],
  AVG(val) val,
  [date]
FROM interval
GROUP BY [name], [date]

```

ORDER BY [date], [name]

Вместо {0} подставляется таблица ShortHistory если дата текущая. Иначе, подставляется таблица LongHistory.

async Task<Dictionary<string, List<Sensor>>> GetIntervalFor(string bDate, string eDate, int poyas)

Предоставляет структуру показаний датчиков в пределах запрошенного интервала по определенному поясу. SQL запрос:

```
select
    o.TagComment name,
    avg(h.VarValue) val,
    format(h.DateTime, 'yyyy-MM-dd') date
from LongHistory h
join Settings_OPCTagsList o
    on h.TagId=o.Tag_ID
    and o.TagName like '%T_Futervki%'
    and o.TagComment like '%пояс {0},%'
where cast(h.[DateTime] as date) between cast('{1}' as date) and
cast('{2}' as date)
group by o.TagComment, format (h.DateTime, 'yyyy-MM-dd')
order by format (h.DateTime, 'yyyy-MM-dd'), o.TagComment
```

Вместо {0} подставляется необходимый номер пояса. Вместо {1 и 2} подставляются необходимые даты в формате ГГГГ-ММ-ДД.

IHistoryRepository

Предназначен для работы с таблицей температур разгара T1150History. Наследует интерфейсу IRepository<T1150History> и расширяет его одним методом.

async Task<Dictionary<string, List<T1150History>>> GetFor(string bDate, string eDate, int poyas)

Метод служит для получения значений температуры разгара по конкретному поясу в пределах необходимого интервала дат. SQL запрос:

```
select *
from T1150History h
where cast(h.[Date] as date) between cast('{0}' as date) and ca
st('{1}' as date)
and h.Poyas={2}
order by h.Luch
```

Вместо {0 и 1} подставляются необходимые даты в формате ГГГГ-ММ-ДД. Вместо {2} подставляется необходимый номер пояса.

ISensorErrorRepository

Предназначен для работы с таблицей ошибок датчиков SensorErrors. Наследует интерфейсу IRepository<SensorError> и расширяет его одним методом.

					90.32.048.ИЗ-02-М	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

async Task<IEnumerable<SensorError>> GetErrors ()

Метод предоставляет перечень всех зарегистрированных ошибок датчиков из таблицы SensorErrors. SQL запрос: **select * from SensorErrors.**

					90.32.048.ИЗ-02-М	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

Контроллер

Серверная часть приложения выполнена в виде контроллера для приема и отправки данных. Контроллер разработан средствами .NET Core 3.1 и реализует интерфейс типового REST api контроллера. Методы в контроллере являются асинхронными, что позволяет добиться большей производительности и избежать необработанных запросов. Используя интерфейсы репозитория (описаны выше) контроллер работает с запросами, отправляемыми на адрес

[http://\[server-name\]/api/heat/\[метод\]](http://[server-name]/api/heat/[метод])

Функции контроллера:

- ❖ Принять GET запрос,
- ❖ В соответствие с запросом обратиться к БД и запросить данные,
- ❖ При запросе на определенный адрес записать новые данные в БД,
- ❖ Вернуть ответ от БД в структурированном виде (формат .json)

Методы контроллера:

async Task<IEnumerable<Sensor>> Instant()

Реализует запрос по адресу <http://ip-computer/api/heat/instant>. Параметры отсутствуют. Выходными данными служат мгновенные минутные данные с датчиков температуры из таблицы Minuta.

async Task<IEnumerable<Sensor>> GetFor(string dt)

Реализует запрос по адресу <http://ip-computer/api/heat/GetFor?dt=...> Параметрами является дата в строковом представлении в формате «ГГГГ-ММ-ДД». Выходными данными являются средние значения 30 минутных данных за выбранную дату из таблицы ShortHistory или LongHistory.

async Task<ActionResult<Dictionary<string, IEnumerable<Sensor>>>> GetFor(string dt, string dt2)

Реализует запрос по адресу

<http://ip-computer/api/Heat/GetForInterval?bDate=...&eDate=...>

Перегруженный метод GetFor для интервала дат. Параметрами являются две даты в строковом представлении в формате ГГГГ-ММ-ДД. Выходные данные – структура показаний датчиков по датам. Применяется в представлении «Разгар горна» при вычислении ошибок датчиков.

async Task<ActionResult<IEnumerable<Sensor>>> GetIntervalFor (string bDate, string eDate, int poyas)

Реализует запрос по адресу

<http://ip-computer/api/Heat/GetIntervalFor?bDate=...&eDate=...&poyas=...>

Параметрами являются даты в строковом представлении в формате «ГГГГ-ММ-ДДДД». Выходными данными являются структуры данных в формате .json,

					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

содержащие дату и набор значений температурных датчиков указанного пояса из таблицы LongHistory.

async Task<ActionResult<IEnumerable<T1150History>>> PullDate(string bDate, string eDate, int poyas)

Реализует запрос по адресу

[http://\[ip-computer\]/api/Heat/PullDate?bDate=...&eDate=...&poyas=...](http://[ip-computer]/api/Heat/PullDate?bDate=...&eDate=...&poyas=...)

Параметрами являются начальная дата, конечная дата и номер пояса. Выходными данными являются структуры данных в формате .json, содержащие дату и набор значений рассчитанной температуры 1150⁰С по 32 лучам указанного пояса из таблицы T1150History.

async Task<ActionResult<int>> PutDate(T1150History[] arr)

Реализует запрос по адресу [http://\[ip-computer\]/api/Heat/PutDate](http://[ip-computer]/api/Heat/PutDate)

Запрос без параметров. Тип запроса Put, в теле запроса присутствует набор рассчитанных значений в формате типа T1150History, готовый для записи в базу данных с помощью запроса для записи, описанного в пункте Работа с таблицами базы данных:.

async Task<ActionResult<IEnumerable<SensorError>>> GetSensorErrors()

Реализует запрос по адресу

[http://\[ip-computer\]/api/Heat/GetSensorErrors](http://[ip-computer]/api/Heat/GetSensorErrors)

Запрос без параметров. Возвращает массив записей таблицы SensorErrors для обработки в представлении.

async Task<ActionResult<int>> PutSensorErrors(SensorError[] arr)

Реализует запрос по адресу

[http://\[ip-computer\]/api/Heat/PutSensorErrors](http://[ip-computer]/api/Heat/PutSensorErrors)

Запрос без параметров. Тип запроса Put, в теле запроса передается массив из всех обнаруженных ошибок датчиков для записи в таблицу SensorError.

async Task<ActionResult<IEnumerable<string>>> GetUsageIps()

Реализует запрос по адресу

[http://\[ip-computer\]/api/Heat/GetUsageIps](http://[ip-computer]/api/Heat/GetUsageIps)

Запрос без параметров. Возвращает список уникальных IP пользователей, когда-либо пользовавшихся функционалом приложения.

async Task<ActionResult<IEnumerable<UsageLog>>> GetUsageFor(string what)

Реализует запрос по адресу

[http://\[ip-computer\]/api/Heat/GetUsageFor?what=...](http://[ip-computer]/api/Heat/GetUsageFor?what=...)

Возвращает список структур UsageLog по указанному в параметре what IP либо дате.

async Task<ActionResult<IEnumerable<UsageLog>>> GetUsageForAll(string dt, string ip)

Реализует запрос по адресу

[http://\[ip-computer\]/api/Heat/GetUsageForAll?dt=...&ip=...](http://[ip-computer]/api/Heat/GetUsageForAll?dt=...&ip=...)

Возвращает список структур UsageLog по указанным в параметрах IP и дате.

					90.32.048.ИЗ-02-М	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

Промежуточные службы (Middleware)

Для регистрации пользователей в таблице посещений (UsageLogs) применен механизм middleware.

Задача middleware LoggingMiddleware - при поступлении запроса к контроллеру перехватить управление и вызвать службу LoggingService.

LoggingService

При поступлении перехваченного запроса служба анализирует его. Для начала, сверяется с конфигурационным файлом приложения, в котором указан список IP, которые не регистрируются в БД. Если IP совпадает, ничего не предпринимать.

Затем, если со времени прошлого идентичного запроса и текущим прошло менее минуты, ничего не предпринимать. Если же запрос новый и его нет в кеше запросов, то сохранить информацию по запросу в базу данных в таблицу UsageLogs используя интерфейс репозитория IUsageLogsRepository.

Основные расчетные функции приложения

Находятся в папке src/calculation. Используются всеми представлениями приложения.

complementData(data)

Входные параметры:

Массив структурированных данных *data*, полученных после обработки начальных данных от контроллера. То есть это значения всех имеющихся температурных датчиков. Так как для расчета необходимы также виртуальные датчики (значения которых получены аппроксимацией соседних значений), эта функция их считает и дополняет массив данных рассчитанными.

Выходные параметры:

Дополненный массив данных.

devideData(data)

Входные параметры:

Массив данных *data*, полученный от контроллера. Функция разбирает наименование датчика и раскладывает значения формируя структурированный массив данных.

Выходные параметры:

Структурированный массив данных.

					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		14

gorn(basicPoyasa, poyasa)

Входные параметры:

Принимает два массива данных. *basicPoyasa* – массив показаний датчиков за начальную дату. *poyasa* – массив показаний датчиков за текущую дату. Задача функции – рассчитать значения температурных изотерм для горизонтальных сечений 7-12 поясов. Расчет 6 пояса осуществляется в функции *leshad()*.

Выходные параметры:

Структурированный массив пар значений (температура-радиус) для всех поясов.

leshad(poyasa)

Входные параметры:

Массив *poyasa* - массив показаний датчиков за текущую дату. Функция рассчитывает высоту температурных изотерм 1150, 800, 500, 300°C для радиусов 0 и 1668 мм, а также на 6 поясе.

Выходные параметры:

Структурированный массив пар значений (температура-высота) для указанных радиусов.

razgarCalcRoot(coefs)

Входные параметры:

Коэффициенты *coefs*, необходимые для расчета корней полинома.

Выходные параметры:

Максимальный корень полинома

smallFunctions

Набор небольших математических расчетных функций, необходимых для расчетов. Состав: *Exp*, *Log*, *Abs*, *Round*, *F_micropor78*, *F_keram80*, *F_polugrafit*, *F_mullit*, *equation* (расчет обычного квадратного уравнения).

Общие классы приложения

Находятся в папке *src/classes*.

Ввиду однотипности расчета изотерм применена цепочка однотипных классов:

Class IHandler

Является абстракцией звена в расчетной цепи. Определяет интерфейс каждого из звеньев. Содержит ссылку на следующий в цепи класс и прототип метода обработки данных.

					90.32.048.ИЗ-02-М	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

class DevideHandler

Наследуется от IHandler и реализует этап разделения данных, т.е. построения структуры температур датчиков по поясам и лучам из информации поступившей от контроллера. Помимо основной обязанности – разделения начальных данных – в зависимости от структуры поступивших данных запускает алгоритм поиска ошибок температурных датчиков.

class ComplementHandler

Наследуется от IHandler и реализует этап дополнения структуры данных температурами данных «виртуальных датчиков», недостающих в информации от контроллера и необходимых для дальнейшего расчета.

class LeshadHandler

Наследуется от IHandler и реализует этап расчета температуры 1150°C лещади и 6 пояса.

class GornHandler

Наследуется от IHandler и реализует этап расчета температур изотерм на поясах 7-12.

class SensorError

Реализует модель «Ошибка датчика». Содержит метод сравнения экземпляров.

class SensorErrorHandler

Инкапсулирует все необходимое для обработки ошибок датчиков. Содержит статический метод handle(), в параметры которого принимает структуру разделенных показаний датчиков. Далее производит сравнение согласно установленному критерию и добавляет ко входной структуре дополнительное поле с найденными ошибками.

Помимо прочего, при обнаружении неучтенной ошибки датчика производит ее запись в базу данных.

Основные графические элементы приложения

NavMenu

Содержит логику работы с титульной строкой приложения. Включает логотип предприятия, заголовок текущего представления, навигационные ссылки.

					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16

Layout

Содержит основное поле приложения, где, в зависимости от выбранной навигационной ссылки отображается то или иное представление.

Footer

Содержит строку подписи приложения.

Представление «Разгар горна»

Функции представления

- ❖ В представлении организовано кеширование данных. Таким образом, ранее запрошенные данные не запрашиваются от контроллера повторно и не просчитываются повторно, а берутся из кэша, что снижает нагрузку на сервер и увеличивает быстродействие приложения.
- ❖ Если кэш не сохранен за требуемую дату, запросить информацию от контроллера и запустить алгоритм расчета (цепочка обработчиков `gornChain`),
- ❖ В зависимости от запроса пользователя предоставить графическое отображение на экране вертикальных или горизонтальных сечений,
- ❖ На отображении текущей даты, если присутствует флаг автоматического обновления – зациклить ежеминутный запрос информации от сервера, отобразить таблицу сравнения данных за текущий и предыдущий часы, отобразить состояние температурных датчиков, наличие ошибок и предупреждения если произошло резкое изменение температуры по какому-либо датчику.

Основные компоненты представления «Разгар горна»

В зависимости от состояния приложения представление отображает необходимые области, такие как:

- ❖ Поле для выбора даты,
- ❖ Меню для выбора горизонтальных или вертикальных сечений,
- ❖ Поле графического отображения выбранного сечения,
- ❖ Таблицу минутных показаний температурных датчиков,
- ❖ Панель неисправностей датчиков,
- ❖ Панель предупреждений о резком изменении температуры.

radialChart

Инкапсулирует логику построения горизонтальных сечений. Имеет внутреннюю функцию *pullPoyas*, которая в зависимости от выбранного пояса фильтрует массив *radiuses*, состоящий из наборов значений (температура-радиус). За отображение графиков отвечает библиотека Amcharts. Фоновые изображения для разных поясов начерчены в масштабе с помощью программы «Компас». Подгонка фона и графика осуществлена средствами css.

					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

verticalChart

Инкапсулирует логику построения вертикальных сечений. Имеет внутреннюю функцию *pullLuch*, которая в зависимости от выбранного луча фильтрует массив *radiuses*, состоящий из наборов значений (температура-радиус). Отображение также происходит благодаря библиотеке *Amcharts*. Содержит также функцию отрисовки индикатора выбранного луча *fillRayIndicator*. Фоновые изображения для разных поясов начерчены в масштабе с помощью программы «Компас». Подгонка фона и графика осуществлена средствами *css*.

class Menu

Инкапсулирует логику меню представления. Предоставляет выбор между горизонтальными и вертикальными сечениями и их лучами.

Алгоритм работы представления «Разгар горна»

- ❖ Начальное состояние – на экране выведено окно с полем выбора даты.
- ❖ Пользователь выбирает дату и нажимает на «Запросить». Если выбранная дата не корректна (выбрана еще не наступившая дата либо недостижимая в базе данных), выдается соответствующее сообщение в виде диалогового окна. Если дата корректна, дата из прошлого, происходит запрос к контроллеру в метод *GetFor* иначе происходит запрос к методу контроллера *GetForTwo* (возвращаются две даты – «сегодня» и «вчера»). Необходимо для расчета вероятной ошибки датчика). Признаком обращения к контроллеру служит появляющееся изображение «Загрузка...». Как только контроллер ответил на запрос изображение исчезает. Контроллер возвращает массив данных в формате *.json*. Этот массив разбирается и структурируется цепочкой обработчиков *IHandler*, формируется массив *radiuses* – все просчитанные значения изотерм по всем поясам плюс лещадь.
- ❖ Появляется меню, в котором можно выбрать необходимое сечение. После выбора сечения, в зависимости от типа необходимого графика вызывается одна из функций *pullPoyas* или *pullLuch*, эти функции формируют массив *chartData*, который передается в компонент *radialChart* или *verticalChart*.
- ❖ Появляется основное поле с графиком, на котором отображается один из компонентов(*radialChart* или *verticalChart*).
- ❖ Если выбрана текущая дата, то на поле графика доступен также переключатель обновления данных, который по умолчанию включен. При необходимости можно отключить обновление. Если выбрана одна из прошлых дат, переключатель не доступен.
- ❖ При обновлении данных запрос автоматически закидывается на 60 секунд. Запрос осуществляется к методу *Instant* контроллера. При этом контроллер запрашивает у базы минутные значения. Изначально загружаются усредненные данные из таблицы *ShortHistory* базы данных, однако день еще не закончен и, поэтому, данные не точны. После того как происходит запрос

					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

минутных значений, их просчет, данные графика меняются и отображаются актуальные минутные значения. Справа от графика появляется таблица с показаниями датчиков для лучшего визуального восприятия. В таблице отображается сравнение температур текущей и средней за предыдущий час.

- ❖ При выборе новой даты работа аналогична описанной выше со 2 пункта.
- ❖ При выборе нового сечения работа аналогична описанной выше с 3 пункта.

Представление «Тренды»

Функции представления

- ❖ Отобразить тренды остаточной толщины стенки печи в пределах запрошенного интервала дат по указанному поясу и лучам.
- ❖ Отобразить тренды температур датчиков в кладке печи в пределах запрошенного интервала дат по указанному поясу и радиусам.
- ❖ Предоставить прогноз остаточной толщины стенки за требуемое количество дней.

Основные компоненты представления «Тренды»

Работа представления организована посредством паттерна «Состояние». Так, все методы обработки запроса инкапсулированы в конкретной реализации абстрактного класса `IStateHandler`. Переключение состояний осуществляется из меню. Меню предоставляет инструмент для выбора необходимых трендов.

class TrendMenu

Инкапсулирует логику работы с меню. Содержит набор типов графиков и инструментов для подготовки критериев выборки данных для отображения необходимой зависимости. Смена обработчика состояния представления происходит при нажатии на один из заголовков меню («Толщина стенки» или «Температура»).

class IStateHandler

Абстрактный класс обработки состояния. Содержит общие необходимые поля для работы с данными и методы, вынесенные из дочерних как общие для них и неизменные от класса к классу. Определяет интерфейс обработчика состояний.

Реализует следующие методы:

- ❖ `handleRadio` – общая часть обработчика нажатия на необходимый пояс
- ❖ `handleCheckbox` – общая часть обработчика нажатия на требуемый луч/датчик
- ❖ `handleDate` – общая часть обработчика нажатия на необходимую дату. В основном определяет механизм защиты от неправильного ввода интервала дат.
- ❖ `clearLuchi` – функция очистки выбранных лучей/датчиков в меню представления

					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

- ❖ *switchHandler* – переключатель обработчика представления, меняющий логику работы представления.
- ❖ *forecastClick* – абстрактный метод. Каждая конкретная реализация должна реализовать этот метод необходимым способом.

class RWallHandler

Наследует классу *IStateHandler* и инкапсулирует логику работы меню температурных трендов остаточной толщины стенки. Реализует следующие методы:

- ❖ *handleRadio* – обработка нажатия на необходимый пояс (радиус лещади)
- ❖ *handleCheckbox* – обработка выбора одного из 32 лучей
- ❖ *clearAll* – очистка компонентов выбора при определенных условиях.
- ❖ *forecastClick* – реализация абстрактной функции базового класса. Запускает алгоритм расчета прогноза толщины стенки на требуемое количество суток. Этот алгоритм реализован в функции *addForecast()*, в которой находятся коэффициенты прямой средних значений температуры по данному лучу за все время работы печи. Зная эти коэффициенты к графику добавляются рассчитанные точки.

Запрос данных происходит в методе *handleRadio*, в котором вызывается функция *handleHistory*, инкапсулирующая логику работы с кэшем состояния, общий алгоритм обработки запрошенного интервала дат. Полученный запрос обрабатывается функцией *makeChartData* (инкапсулирует логику формирования данных в необходимом для построения графика формате).

class TempHandler

Наследует классу *IStateHandler* и инкапсулирует логику работы меню температур датчиков. Реализует следующие методы:

- ❖ *handleRadio* – обработка нажатия на необходимый пояс (радиус лещади)
- ❖ *handleCheckbox* – обработка выбора одного из 32 лучей
- ❖ *clearAll* – очистка компонентов выбора при определенных условиях.
- ❖ *hideChBoxes* – скрывает контейнер с лучами на время загрузки данных и перестроения входящих в пояс лучей.
- ❖ *showChBoxes* – отображает контейнер с загруженными и перестроенными лучами.

Запрос данных происходит в методе *handleRadio*, в котором вызывается функция *handleTemp* (осуществляет запрос на сервер, производит построение структуры ответа). Затем, при щелчке на нужном радиусе луча происходит перестроение структуры *chartData* в методе *makeChartData*, реагируя на которое изменяется график основной области представления.

class R1150Chart

Служит для графического отображения выбранной зависимости. На вход принимает структурированные подготовленные данные *chartData*, при изменении которых перерисовывает график с учетом поступивших изменений. Тип графика – точечный с соединенными точками.

					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

class Trends

Основной класс компонента. Ввиду различной логики отборки критериев для построения графиков, содержит внешние обработчики его работы, выделенные в отдельные сущности – обработчики состояния. Таким образом, выбирая другой тип графика в меню изменяется логика работы с данными у родительского класса Trends соответственно. Обработчики состояний представлены отдельными классами, инкапсулирующими всю необходимую логику для обработки данных.

Алгоритм работы состояния rWall

- ❖ Изначально в интервале дат выбраны 2 недели, которые могут быть расширены при желании пользователем. Доступно меню, состоящее из полей выбора необходимых лучей (32 штуки) и поясов.
- ❖ Для запроса к контроллеру необходим интервал дат и пояс, а для отображения графика необходим набор датчиков на конкретных радиусах выбранного луча. Запрос происходит в момент нажатия на пояс.
- ❖ Когда ответ контроллера получен, представление сохраняет данные в кэше, чтобы сократить время обработки при аналогичном последующем запросе.
- ❖ Справа от поля выбора пояса появляется подменю с выбором необходимых датчиков на требуемом луче. Подменю динамически реагирует на запросы.
- ❖ После выбора нужного датчика происходит окончательная обработка структуры данных полученной от контроллера и функцией *makeChartData*, изменяется массив в формат, необходимый библиотеке *amcharts* для построения графика.
- ❖ Появляется поле с графиком.
- ❖ При выборе нового датчика (отмене одного из отображенных), обработчик *handleCheckbox* пересоздает имеющуюся структуру данных для графика с учетом изменений и график реагирует на это изменением отображаемых кривых.
- ❖ При смене пояса происходит описанная выше процедура получения информации для другого пояса.
- ❖ Аналогично при смене дат.

Алгоритм работы состояния Temp

- ❖ Изначально в интервале дат выбраны 2 недели, которые могут быть расширены при желании пользователем. Доступно меню, состоящее из полей выбора необходимых поясов.
- ❖ Для запроса к контроллеру необходим интервал дат и пояс, а для отображения графика необходим набор датчиков на конкретных радиусах выбранного луча. Запрос происходит в момент нажатия на требуемый пояс.
- ❖ Когда ответ контроллера получен, представление сохраняет данные в кэше, чтобы сократить время обработки при аналогичном последующем запросе.

					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		21

- ❖ Справа от поля выбора пояса появляется подменю с выбором необходимых датчиков на требуемом луче. Подменю динамически реагирует на запросы.
- ❖ После выбора нужного датчика происходит окончательная обработка структуры данных полученной от контроллера и функцией *makeChartData*, изменяется массив в формат, необходимый библиотеке *amcharts* для построения графика.
- ❖ Появляется поле с графиком.
- ❖ При выборе нового датчика (отмене одного из отображенных), обработчик *handleCheckbox* пересоздает имеющуюся структуру данных для графика с учетом изменений и график реагирует на это изменением отображаемых кривых.
- ❖ При смене пояса происходит описанная выше процедура получения информации для другого пояса.
- ❖ Аналогично при смене дат.

Представление «Статистика»

Функции представления

- ❖ Визуально отобразить статистику посещения сайта по запрошенному IP пользователя либо за определенную дату
- ❖ Визуально отобразить статистику посещения сайта по двум параметрам (IP и дата)

Основные компоненты представления

Class Usage

Реализует базовый компонент представления. Содержит методы для отображения таблицы с информацией по посещениям. Содержит обработчики нажатий на запросы по определенным критериям. Содержит меню для формирования запроса

Class UsageTable

Реализует компонент отображающий основную информацию по посещениям.

Алгоритм работы представления

- ❖ По нажатию на одну из кнопок меню формирования запроса выполнить асинхронный запрос к контроллеру,
- ❖ По поступлению информации от контроллера сформировать удобную для восприятия структуру данных (заменить веб-представление понятным человеку),
- ❖ Передать полученную структуру компоненту *UsageTable* для отрисовки.

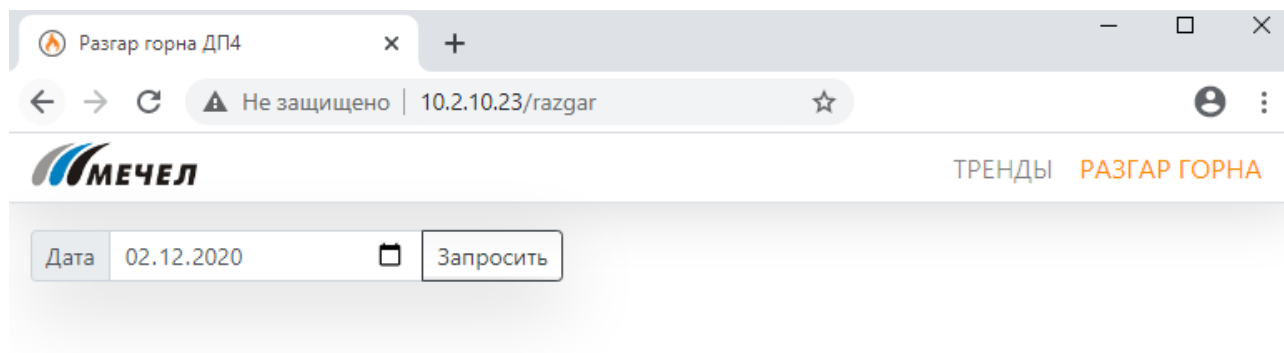
					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

4. СООБЩЕНИЯ

Проверка работы программы

Начальная страница

После запуска сайта необходимо проверить работу приложения, открыв браузер и перейдя по адресу **http://[ip-address]**. Начальная страница сайта состоит из одного поля – Дата.



4.1 – Внешний вид исправного сайта

Сайт должен отвечать на запросы и предоставлять необходимую информацию без длительных задержек. Подробно интерфейс и функциональность приложения описана в *T2.K1.Руководство пользователю*.

Страница температурных трендов

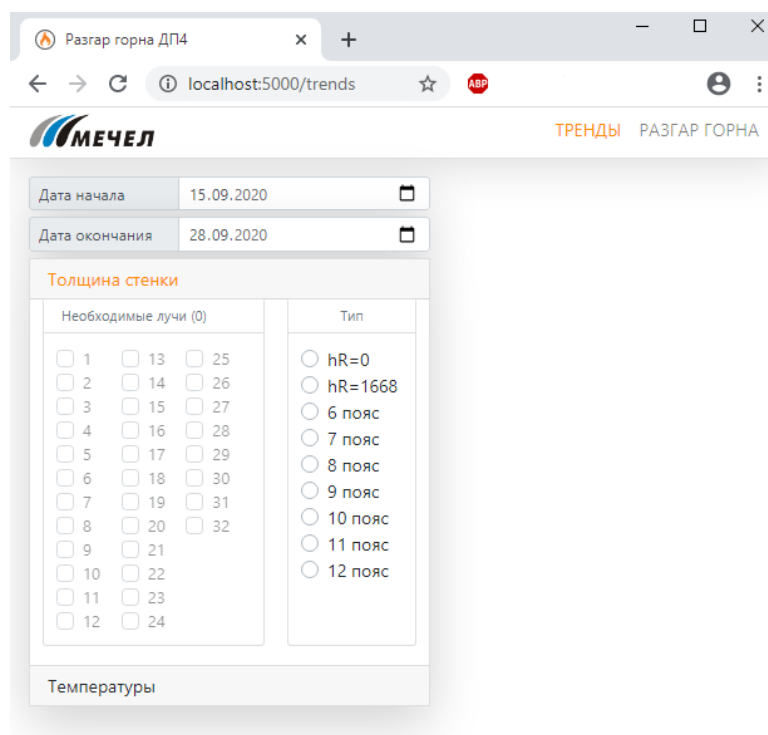


Рисунок 4.2 – начальная страница для работы с температурными трендами

Запрос не существующей страницы

При ручном вводе несуществующего адреса в адресную строку браузера должно отобразиться соответствующее сообщение:

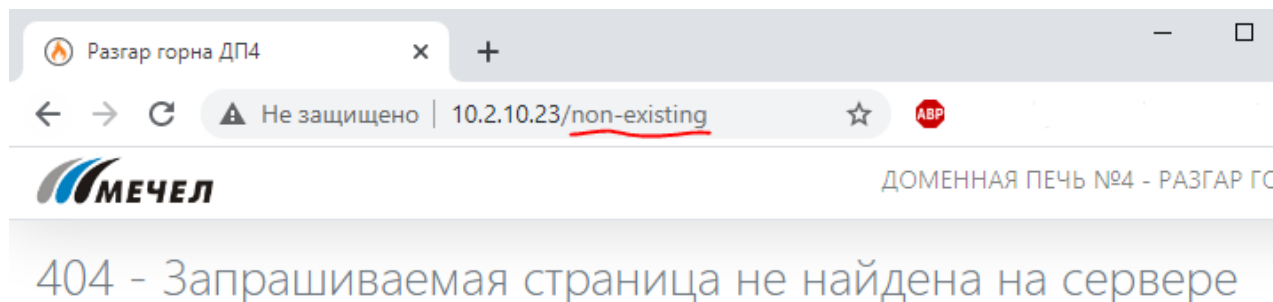


Рисунок 4.3 – Запрос не существующей страницы

Проверка работы контроллера

Для проверки работоспособности контроллера необходимо выполнить запрос на адрес сервера следующего вида:

[http://\[ip-address\]/api/heat/instant](http://[ip-address]/api/heat/instant)

В ответ сервер должен предоставить информацию в формате .json

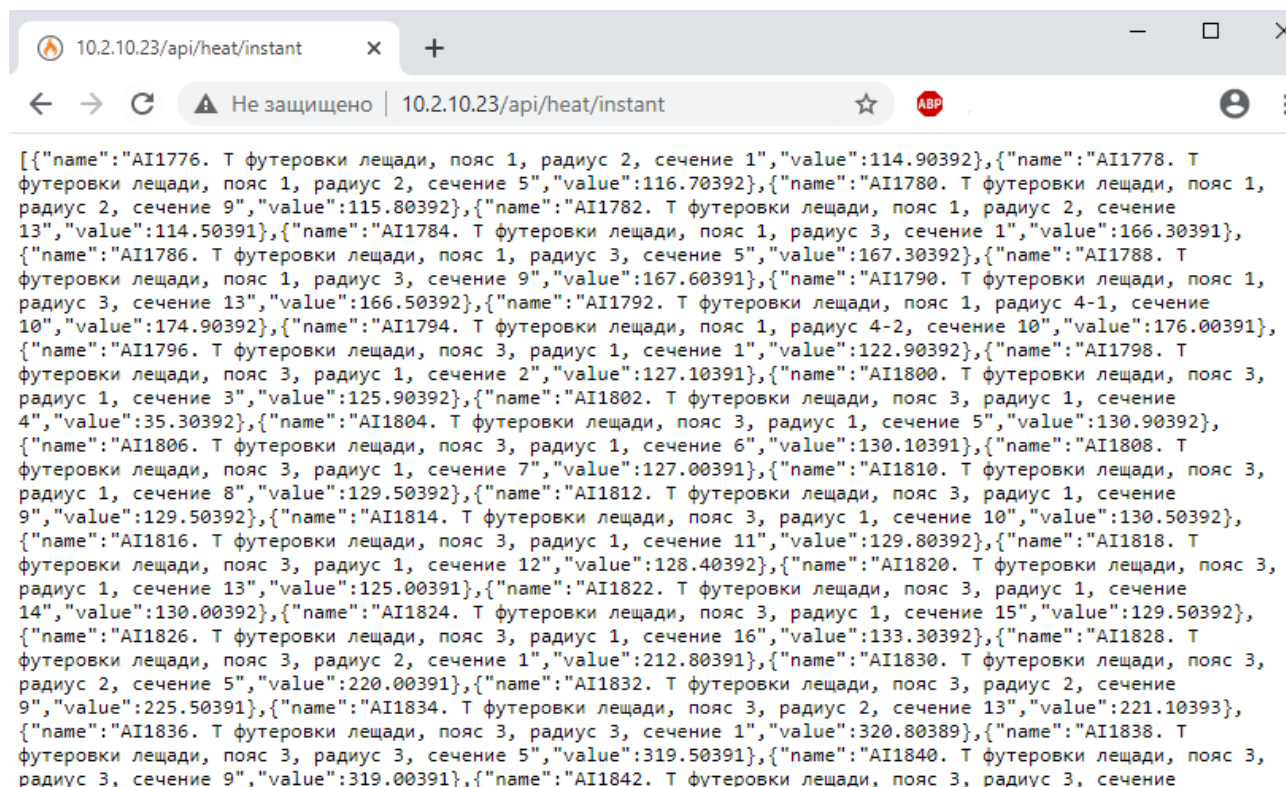


Рисунок 4.4 – ответ от контроллера в формате .json

					90.32.048.ИЗ-02-М	Лист
Изм.	Лист	№ докум.	Подпись	Дата		24

Запрос не существующей даты

При запросе не существующей даты отображается предупреждение

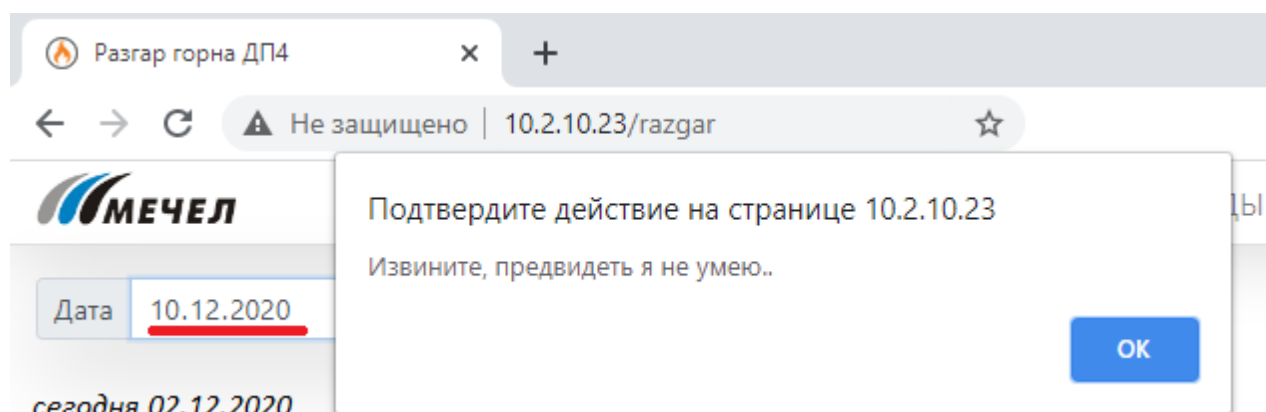


Рисунок 4.5 – Запрос неправильной даты

Запрос даты вне информационной доступности

При запросе даты до 15.02.2020 (дата настройки оборудования ДП4 после кап. ремонта) также отображается предупреждение.

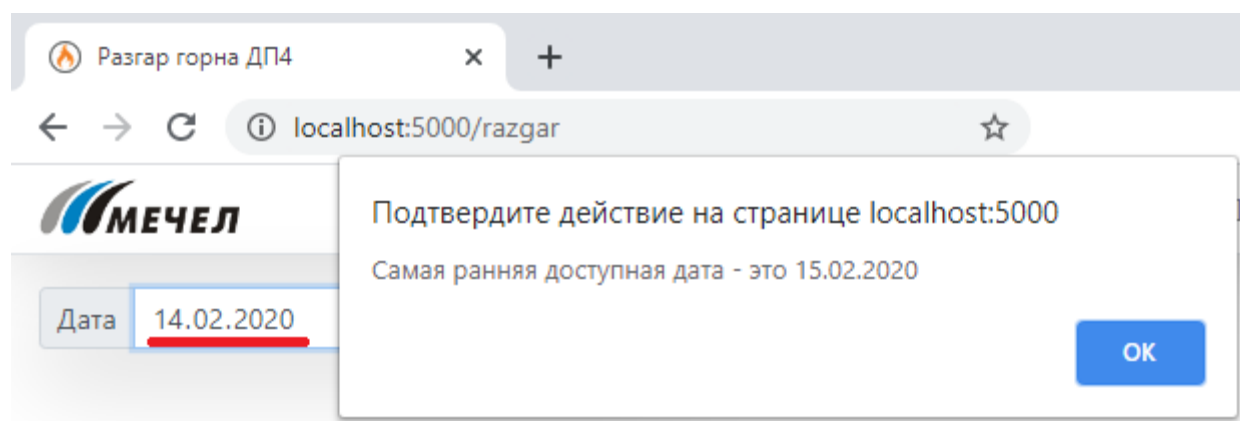


Рисунок 4.6 – Запрос даты вне информационной доступности

Перечень возможных неисправностей

Не исчезает значок Загрузка и не появляется меню

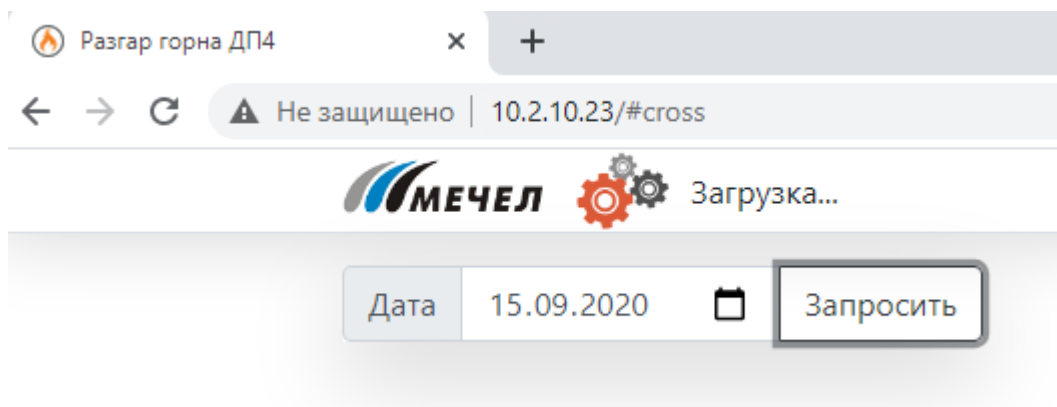


Рисунок 4.7 – Неверная строка подключения

- ❖ Неверная строка подключения. Необходимо проверить корректность строки подключения в конфигурационном файле приложения:

[корень-проекта]\ bin\Release\netcoreapp3.1\win10-x64\publish\appsettings.json

- ❖ Отсутствует общий доступ к папке релиза проекта (физический путь сайта). Необходимо дать права на доступ для «Все» к этой папке.

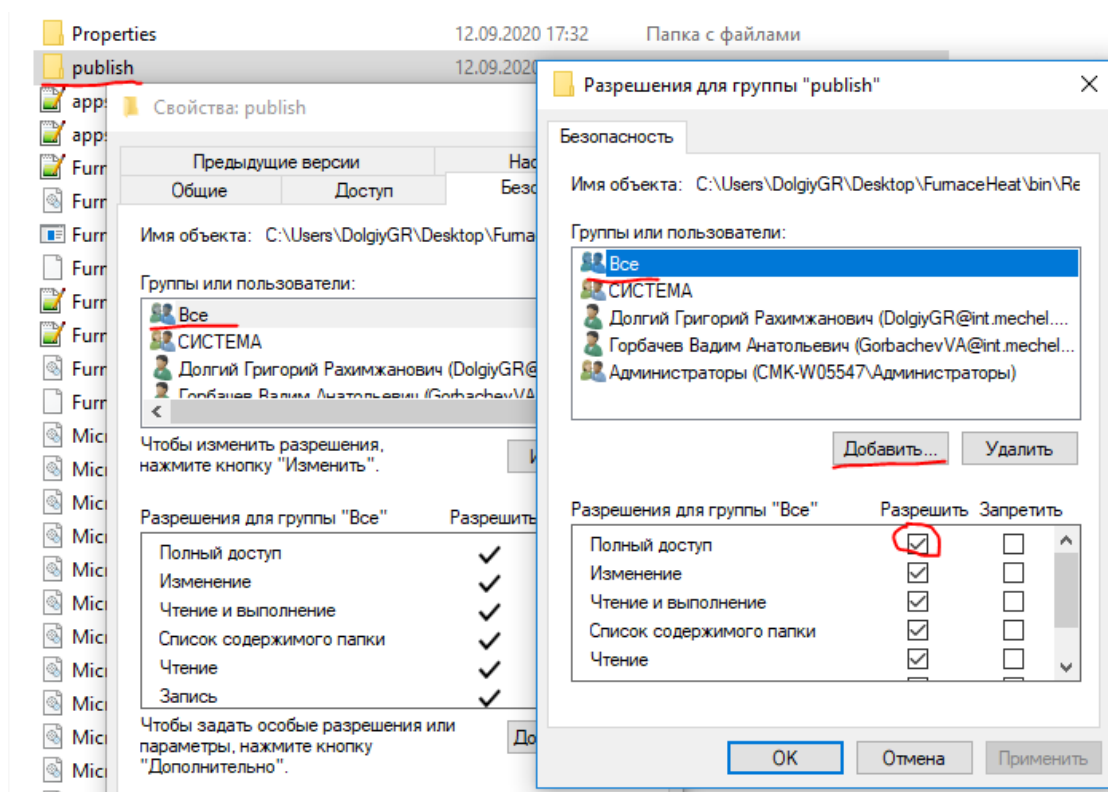


Рисунок 4.8 – Разрешение на полный доступ

Не удается получить доступ к сайту

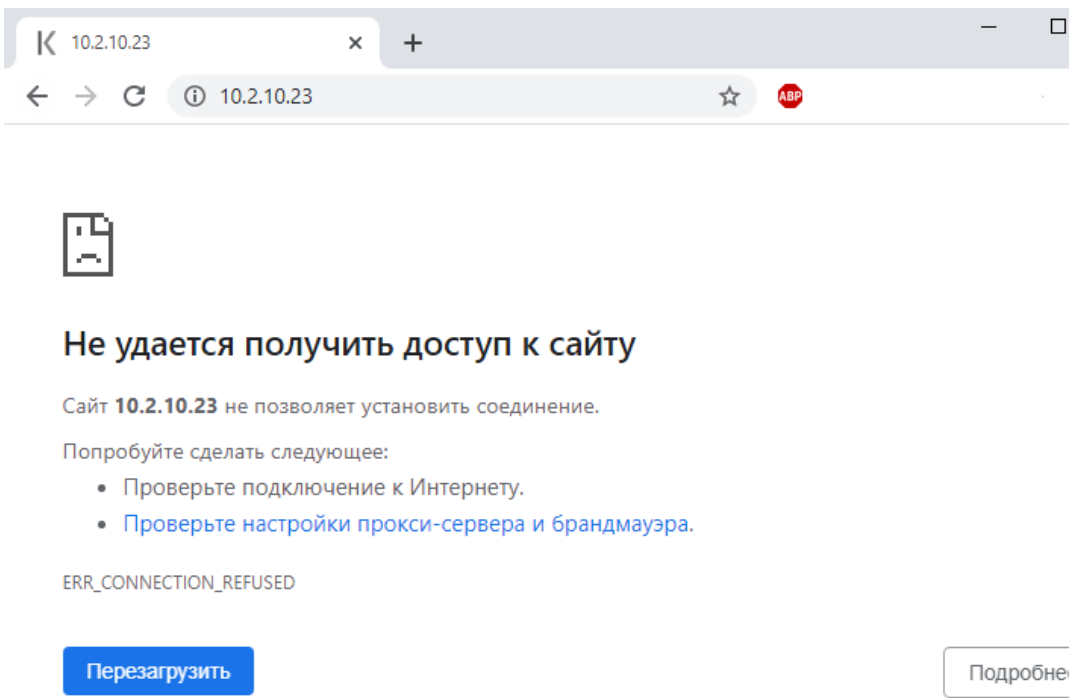


Рисунок 4.9 – Нет доступа к сайту

- ❖ Для запуска сайта необходимо открыть Диспетчер служб IIS и проверить визуально работоспособность сайта (нет значка Стоп). При необходимости запустить сайт нажатием соответствующей кнопки на правой панели.
- ❖ Также необходимо проверить доступность машины средствами командной строки например. Для проверки запустить командную строку и выполнить команду `ping [ip-компьютера]`. Если пинга нет, то проверить правила Windows Firewall или другого установленного на данном компьютере.

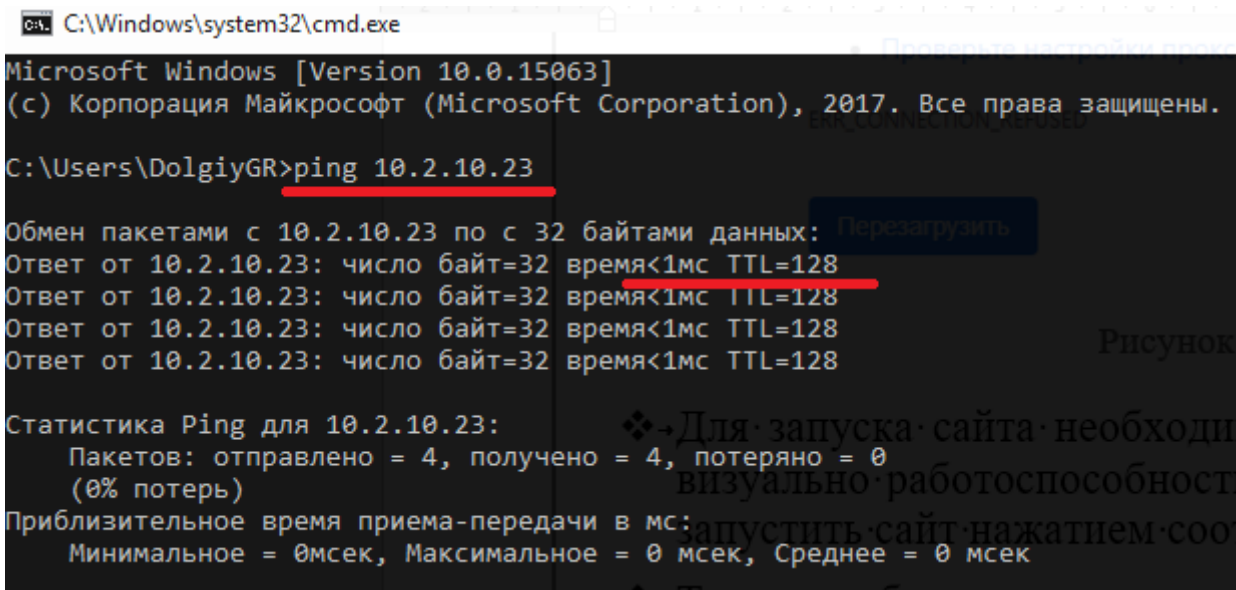


Рисунок 4.10 – проверка доступности сервера

- ❖ Если используется браузер MS Edge (версия ниже либо равная 40.15063.0.0), необходимо включить поддержку синтаксиса ES6 javascript. Для этого в адресной строке необходимо набрать **about:flags** и нажать enter.

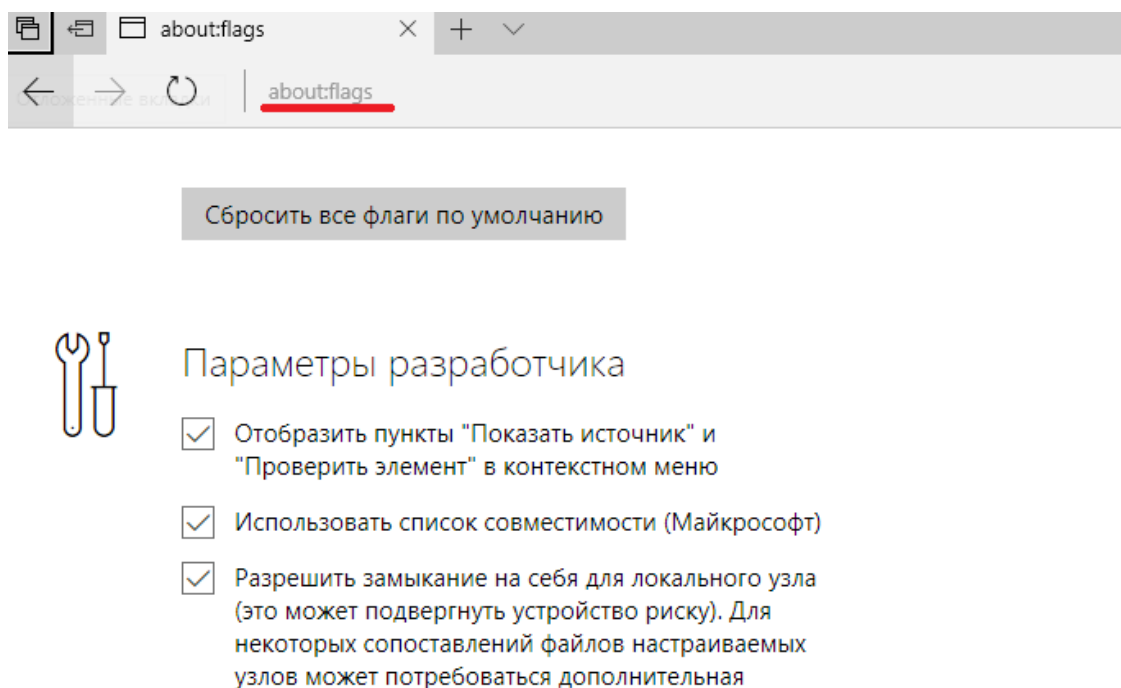


Рисунок 4.11 – Меню разработчика MS Edge

Далее необходимо прокрутить вниз до пункта JavaScript и включить галочку «Включить экспериментальные возможности JavaScript». Затем перезапустить браузер и страница должна отобразиться.

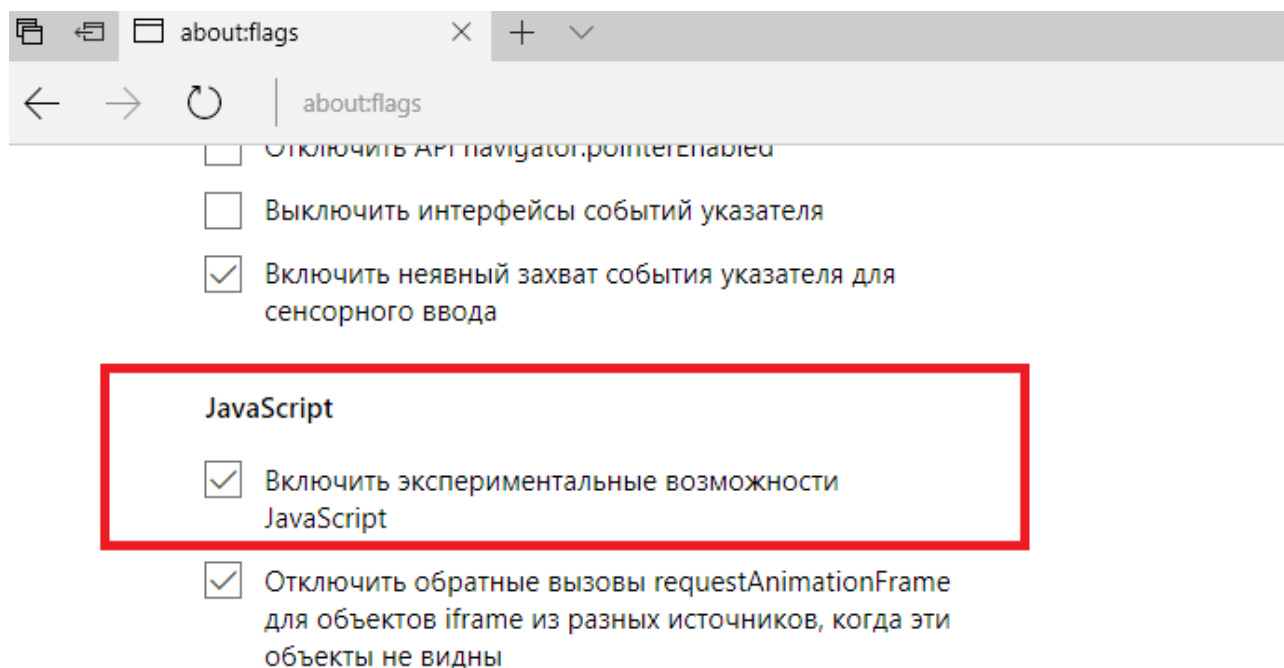


Рисунок 4.12 – Включение синтаксиса ES6