Веб-сайт состоит из двух основных частей: серверной части и веб-интерфейса, серверная часть отвечает за вызовы сервера и API и поддерживает функциональность веб-сайта. Веб-интерфейс – это часть, которая взаимодействует с пользователем и которая соединяет пользователя с сервер. Файловая иерархия состоит из файла server.js, который содержит весь основной код на стороне сервера и вызовов API, и файла package.json, Одним из наиболее важных файлов является файлом Package.json, который содержит информацию о приложении, зависимости и скриптах.

## 2.1. Внешний интерфейс

До сих пор мы работали с английской версией API, по сути, то же самое делается с российской версией, за исключением того, что она использует разные вызовы API для российских источников.

Код интерфейса написан на Pug.js, который является посредником.

Pug – это шаблонизатор для Node.js. Шаблонный движок позволяет вводить данные и затем генерировать HTML. Вкратце: во время выполнения Pug заменяет переменные в файле действительными значениями, а затем отправляет полученную строку HTML клиенту.

Первые две важные кнопки на веб-сайте – это кнопки изменения языка,

которые переключаются между русской и английской версиями веб-сайта, в файле сервера пути уже заданы для двух страниц, а код pug просто указывает браузеру правильный путь.

Листинг 1

Код для кнопок смены языка в Pug.js

. btn*−*group . btn*−*group*−*l g ( s t y l e=’width:100%;padding *−*bottom: 1 0px;’) a . butto n . btn . btn*−*b l o c k . b t n s t y l e ( ty pe=’butto n’, s t y l e=’bo rde r *−* r a d i u s:10px10px10px10px;’, t a b i n d e x=’3’h r e f="/E n g l i s h")

E n g l i s h

a . butto n . btn . btn*−*b l o c k . b t n s t y l e ( ty pe=’butto n’, s t y l e=’bo rde r *−*

r a d i u s:10px10px10px10px;’, t a b i n d e x=’4’h r e f="/Rus s i an") Ru s s i an

1

2

3



Рисунок 3. Кнопки смены языка

Далее идет область, включающая в себя два основных компонента: текстовое поле и кнопку отправки. Первое включает в себя заданные команды, которые не позволяют пользователю использовать несколько слов даже при копировании и вставке в текстовое поле, чтобы избежать осложнений, поскольку веб-сайт создан для того, чтобы давать определения отдельным словам. Когда пользователь нажимает кнопку «Отправить», используя POST, слово отправля- ется на внутренний сервер для соответствующей обработки[5].

Листинг 2

Форма для поиска слова

form ( a c t i o n=’s e arc hEn’, method=’POST’, s t y l e=’h e i g h t: 9 7px;width: 1 0 0 % ; f l o a t:r i g h t;’)

i n p u t . btn . btn*−*l g . btn*−*rounded . s e a r c h b u t t o n ( ty pe=’submit’, t a b i n d e x= ’6’o n c l i c k="f u n c 1( )")

i n p u t#n o s p ac e ( ty pe=’t e x t’, name=’o x f o r d’, s t y l e=’padding:29px32 px;width: 6 9 % ;h e i g h t: 1 0 0 % ;f l o a t:l e f t;d i s p l a y:i n l i n e *−*b l o c k; f o n t *−*s i z e: 4 0px’, t a b i n d e x=’5’)



Рисунок 4. Кнопка отправки

После этого веб-сайт разделяется на столбцы для каждого источника словаря. Pug.js поддерживает интерполяцию, которая позволяет серверу отправлять данные, включая строку и HTML-код, во внешний интерфейс, используя «!» Для строки HTML и «» для обычной строки, используя Функция res.render на сервере, например столбец Оксфордского словаря:

Листинг 3

Колонка для Оксфордского словаря

h2 ( tabindex=’7’) Oxford Dictionary

butto n . btn . btn*−*sm . copybtn ( onclick="copyTo Clipboard( ’#OxDic’ )", tabindex=’8’) Copy Text

h3#OxDic ( t a b i n d e x=’9’) p#Oxformat #{idword } p ! { oxforddefinition }

p ! { oxexample }

Также для простоты использования кнопка копирования была добавлена к каждому источнику для быстрого копирования информации.

Листинг 4

Функция копирования

f u n c t i o n copyTo Clipboard ( e l e m e n t ) { v ar $temp = $ ("<i nput>") ;

$ ("body") . append ( $temp ) ;

$temp . v a l ( $ ( e l e m e n t ) . t e x t ( ) ) . s e l e c t ( ) ; document . execCommand ("copy") ;

$temp . remove ( ) ;

}

## 2.2. Серверная сторона

Веб-сайт написан с использованием трех основных технологий: Node.js для серверной части, Express.js, который представляет собой инфраструктуру Node.js, и Pug.js, который является механизмом шаблонов. Далее представлены краткие определения этих технологий.

### 2.2.1. Среда выполнения Node.js

Node.js – это межплатформенная среда выполнения JavaScript, которая может использоваться как для серверов, так и для настольных приложений. Она может использоваться для создания различных типов приложений, таких как приложение командной строки, веб-приложение, приложение чата в реальном времени, Сервер API REST и т.д. Однако он в основном используется для создания сетевых программ, таких как веб-серверы.

Node.js предназначен для работы на выделенном HTTP-сервере и использования одного потока с одним процессом за раз. Приложения Node.js основаны на событиях и работают асинхронно. Код, построенный на платформе Node, не соответствует традиционной модели приема, обработки, отправки, ожидания, получения. Вместо этого Node обрабатывает входящие запросы в постоянном стеке событий и отправляет небольшие запросы один за другим, не ожидая ответов.

Следует отметить, что Node.js не является JavaScript сам по себе, это программа для запуска JavaScript. JavaScript не справляется с сокетами уровня операционной системы и проблемами с сетевым

подключением. Программа Node написана на языке C, который способен выполнять мощные функции. JavaScript идеально подходит для отправки инструкций в C-программу, которая может быть выполнена в глубоких слоях ОС.

Это отступление от базовых моделей, которые запускают более крупные и более сложные процессы и одновременно запускают несколько потоков, каждый из которых ожидает своего соответствующего ответа, прежде чем двигаться дальше.

### 2.2.2. Фреймворк Express

Express – это веб-инфраструктура маршрутизации и промежуточного программного обеспечения, которая обладает минимальной собственной функциональностью: приложение Express – это, по сути, серия вызовов функций промежуточного программного обеспечения, действующая только как тонкий слой основных функций веб-приложений. В Express нет встроенного механизма реляционного сопоставления объектов или шаблонов. Он не построен на конкретных компонентах, он стремится передать управление в руки разработчика и упростить разработку веб-приложений для Node.js. Express наиболее популярен среди стартапов, которые хотят создать продукт как можно быстрее и не имеют большого количества унаследованного кода.

### 2.2.3. Как работает серверная часть

Основным файлом, который запускает серверную часть веб-сайта, является server.js, который начинается с объявления всех используемых модулей и зависимостей:

Листинг 5

Инициализация библиотек Node.js

|  |  |  |
| --- | --- | --- |
|  | v | express = require (’express’) ; |
|  | v | app = express ( ) ; |
|  | v | request = require (’request’) ; |
|  | v | https = require (’https’) ; |
|  | v | Dictionary = require (“oxford*−*dictionary*−*api”) ; |
|  | r | path = require (’path’) ; |

Включая express.js, который является используемой платформой. Модуль запроса, который позволяет серверу отправлять вызовы API, а также собственный Оксфордский API node.js. Затем мы начнем отображать пути к файлам для веб-сайта, у которого есть две стороны – русская и английская – учитывая, что веб-сайт будет работать на двух языках, которые требуют разной компоновки и разных вызовов API. Такой метод оказался намного быстрее, чем создание двух отдельных страниц для каждой версии.

По умолчанию интерфейс веб-сайтов написан на HTML, так как онлайн-словарь будет использовать Pug.js для рендеринга в HTML, поэтому для Механизм просмотра (View engine) необходимо установить Pug, и установить путь к общедоступной папке, которая содержит ресурсы веб-сайта, включая изображения, файлы CSS и сценарии.

Листинг 6

Настройка сервера

1

app . s e t (’viewe n g i ne’,’pug’) ; app . u s e ( exp ress .static(

path . join ( \_\_dirname ,’public’) ) ) ;

2

Команда app.get используется для установки путей с целью навигации по веб-сайту, поэтому на интерфейсных кнопках нужно просто вызывать эти пути, чтобы отправить пользователя на нужную страницу.

Листиг 7

Настройка путей для страниц сайта

1

app . g e t (’/’, f u n c t i o n ( req , r e s ) { r e s . r e n d e r (’MainEn’) ;

} ) ;

app . g e t (’/Rus s i an’, f u n c t i o n ( req , r e s ) { r e s . r e n d e r (’MainRus’) ;

} ) ;

app . g e t (’/E n g l i s h’, f u n c t i o n ( req , r e s ) { r e s . r e n d e r (’MainEn’) ;

} ) ;

2

3

4

5

6

А теперь та часть, где пользователь нажимает кнопку отправки после заполнения текстового поля: сначала он извлекает слово, которое было отправлено using the POST method.

Листинг 8

Получение слова со стороны пользователя

app . p o s t (’/s e arc hEn’,

function ( searchReq , searchRes ) {

v ar searchInput ="";

var searchInputVal ="";

s e arc h Re q . on (’data’, f u n c t i o n ( data ) { searchinput += data ;

console.log ("s e a r c h I n p u t"+ s e a r c h I n p u t ) ;

searchInput = searchInput.split ("=") ;

seachInputVal = searchInput [ 1 ] ;

console.log ("searchInputVal” + searchInputVal ) ;

1

2

3

4

5

6

7

Т.к. Node.js написан на Javascript, это означает что он асинхронизировано, то есть текущая выполняемая функция не блокирует поток, а программа не ожидает результата, а переходит к следующему шагу во время выполнения предыдущей функции, чтобы убедиться, что все три вызова API были сделаны и завершены до работы с результатами. В данном проекте был использован модуль под названием «Async», который позволяет нескольким функциям работать параллельно и позволяет программе двигаться вперед, только если все они были завершены.

Листинг 9

Функция Async.Parallel

async . parallel ({

one : function ( cb ) {

\\ Первый вызов API

} ,

two : function ( cb ) {

\\ Второй вызов API

} ,

1

После этого каждый вызов возвращает объект JSON, содержащий требуемые данные. Следующим шагом является их обработка и форматирование и проверка того, что обрабатываются ситуации, когда результат не найден или ввод недействителен.

Листинг 10

Форматирование строки JSON

1

i f(yantr.includes(’"d e f" : [ ]’) ) {

yantre="No word Available";

}e l s e{

\\ format string }

i f( oxstr.includes ("No entry available for")

| oxtr == ( undefined ) { o x d i c="No word Available";

}e l s e{

\\ format string }

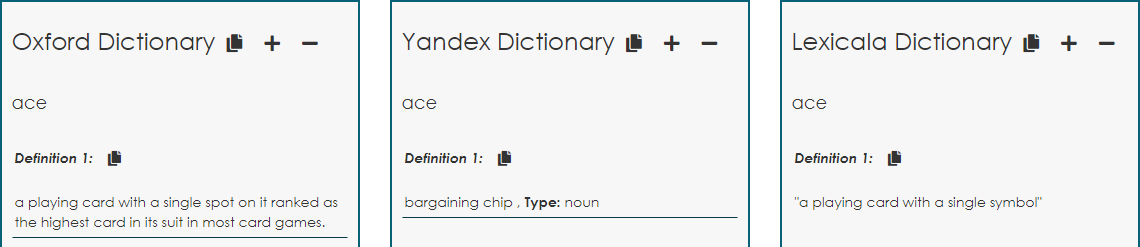


Рисунок 5. Результаты представления

До сих пор мы работали с английской версией API, но по сути, то же самое делается с российской версией, за исключением того, что она использует разные вызовы API для русских источников.

Листинг 11

Отправка результатов на сайт

s e a r c h R e s . r e n d e r ("MainEn", { yantitle : yantxt ,

yanposition : yanpos ,

yanspeech : yants ,

yantrrans : yantr ,

idword : oxdic } ) ;

1

2

3

4