УДК 004.021

ВЫБОР ТЕХНОЛОГИЙ И РАЗРАБОТКА АДАПТИВНОГО ВЕБ-ПРИЛОЖЕНИЯ «ИНФОРМАЦИОННЫЙ АГРЕГАТОР БАНКОВ РЕСПУБЛИКИ БЕЛАРУСЬ»

Р.П. ГУСЕВСКИЙ
(Представлено: Д.В. ПЯТКИН)

Представлен практический способ создания интерфейса для веб-приложения «Информационный агрегатор банков Республики Беларусь». Целью работы явилось написание веб-приложения с удобным интерфейсом, который агрегирует данные с различных официальных банковских сайтов и предоставляет общие сведения по курсам валют в виде графиков, а также новостям и предоставляет ссылки на нужную информацию о банках, информация о местоположениях банкоматов, отделений и обменных пунктов.

Современному человеку представить жизнь без банка очень сложно. Сегодня через банки проходит практически вся наша финансовая жизнь: мы получаем зарплату на банковские карты, берем кредиты, ипотеку, платим через банки за образование детей, оплачиваем услуги ЖКХ, получаем и отправляем денежные переводы, обмениваем денежные средства. Из-за того, что данная тематика всегда будет актуальна, выбрана разработка веб-приложения «Информационный агрегатор банков Республики Беларусь». Немалую роль играют веб-приложения агрегаторы, где консолидируются сведения одновременно из нескольких источников. Они представлены в формате единого пользовательского интерфейса. Такого плана ресурсы крайне полезны при поиске наиболее выгодных условий предоставления услуг и быстрого получения всех необходимых данных. Приложения-агрегаторы – это правильный выбор, когда необходимо быстро найти искомое или сориентировать пользователя на целевые действия [1].

Разработка и выбор технологий дляданного веб-приложения

Для реализации приложения были выбраны языки программирования: Javascript, Typescript, программная платформа NodeJS, фреймворк Angular 2, языком разметки веб-страниц является HTML, а также применялись каскадные таблицы стилей (CSS).

Node.js представляет среду выполнения кода на JavaScript, которая построена на основе движка JavaScriptChrome V8, который позволяет транслировать вызовы на языке JavaScript в машинный код. Node.js прежде всего предназначен для создания серверных приложений на языке JavaScript [2].

Для подключения MongoDB к NodeJS использовался фреймворк Express и система управления БД MongoDB для Windows Server 2008 R2 64-bit andlater, with SSL support x64.

Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Express позволяет иметь в своем распоряжении множество служебных методов HTTP и промежуточных обработчиков, что позволяет создать надежный API как можно быстрее и легче. Рассматриваемый фреймворк обладает хорошей производительностью за счет тонкого слоя фундаментальных функций веб-приложений, которые не мешают вам работать с функциями Node.js [3].

Чтобы установить связь между API и базой данных, хранящейся в MongoDB, нужно воспользоваться MongoDB Database Server, который устанавливается вместе с системой управления. Для хранения строки соединения с базой данных был создан отдельный модуль db.js (листинг 1).

Листинг 1 – модуль db.js

```
varMongoClient = require('mongodb').MongoClient;

var state = {
db: null
};

exports.connect = function(url, done){
  if(state.db){
     return done();
  }
MongoClient.connect(url,function (err,db){
if(err){
       return done(err);
   }
```

```
state.db = db;
     done();
  })
};

exports.get = function(){
   return state.db;
};
```

Для лучшей работы с БД разделена логика на модель и контроллер. Создано два модуля: один модуль banks.js, который лежит в каталоге controllers, а другой модуль с аналогичным названием, который располагается в каталоге models.

В модели лежит основная логика для работы с базой данных в MongoDB, а в контроллере – логика для обработки запросов на сервер и получения информации пользователю, который использует данное API. Исходя из того, что модель и контроллер связан, будет предоставлен листинг с работой одного метода как в модели (листинг 2), так и в контроллере (листинг 3).

Листинг 2 – метод получения информации банка в модели

```
vardb = require('../db');

exports.getBankByShortName = function(shortName,cb){
db.get().collection('banks').findOne({ shortName: shortName}, function (err,doc) {
cb(err,doc);
   });
};
```

Как описывалось ранее, логика контроллера напрямую зависит от логики модели, что описывается в примере, указанном ниже (листинг 3). Здесь установлена связь при помощи метода require, который принимает путь модуля и возвращает переменную-связь с данным унаследованным модулем. Require дает знать, что данный модуль наследует весь функционал из предыдущего модуля, обозначенный ключевым словом exports.

Листинг 3 – метод получения информации банка в контроллере

```
var Banks = require('../models/banks');
exports.getNameBanks = function(req,res){
  Banks.all(function (err, docs){
            if(err){
            console.log(err);
            return res.sendStatus(500);
            }
            docs = docs.map(function(bank){
            return {nameBank : bank.nameBank, shortName: bank.shortName ,pathImg: bank.pathImg};
            });
  res.send(docs);
  });
};
```

На клиентской части использовался фреймворк Angular 2. Он предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и многое другое [4]. Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript, что хорошо помогает в тестировании отдельных модулей и компонент приложения, так как этот язык, в отличие от Javascript, является строго типизированным.

Для того чтобы создать приложение на Angular 2, нужно установить Angular CLI, данный пакет поможет быстро развернуть и настроить проект. После разворачивания проекта необходимо создать модуль приложения, создать компоненты, разметку и стили для компонент и приступать к работе.

Весь дизайн сайта написан вручную, для достижения адаптивной верстки использован bootstrap 3.

При разработке дизайна клиентской части веб-приложения были учтены следующие *правила*:

1. Использование высококачественных изображений.
2. Существование каждого отдельного элемента на веб-странице должно быть чем-то обусловлено.
3. Изучение сайтов-аналогов.
4. Простота конструкции.
5. Подбор шрифта в сочетании с дизайном, шрифт одна из важнейших частей дизайна.
6. Больше графики, меньше текста.
7. Выделение важной информации.
8. Оптимизация сайта для удобства пользователя.
9. Умение продумывать и заполнять «воздушное пространство» на веб-сайте.
10. Краткая подача материала.

Главная страница с панелью навигацией показана на рисунке 1.

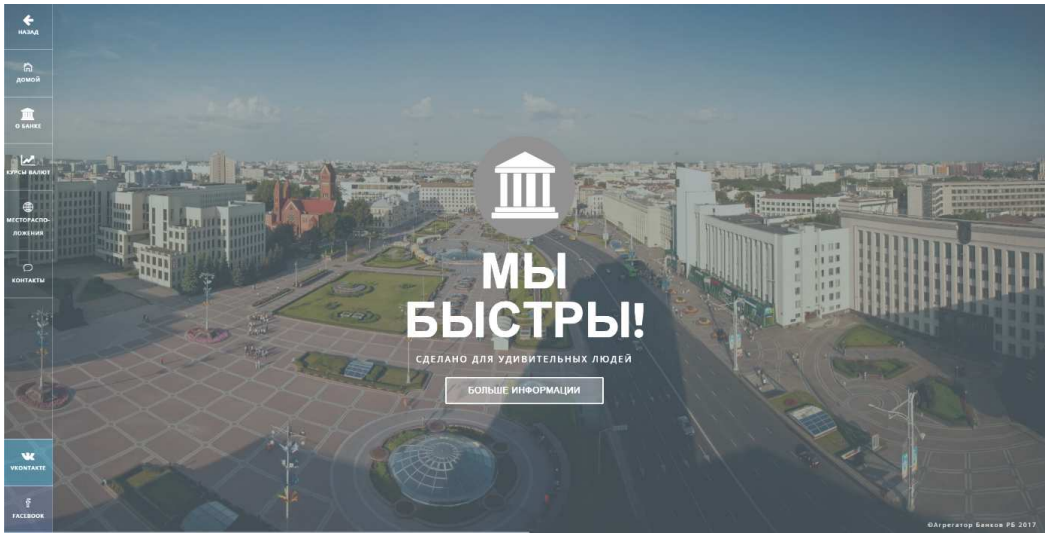

Рисунок 1. – Главная страница с панелью навигацией

Заключение

Авторами разработано веб-приложение с адаптивным дизайном, с серверной частью, которая была хорошо спроектирована и разделена на сущности контроллера и модели, что позволяет пользователю осуществить поиск нужной информации о банке в короткий промежуток времени. Надо всего лишь выбрать нужный банк и на панели навигации выбрать пункт, который нужен пользователю. На данный момент ведутся работы по увеличению функционала программы.

ЛИТЕРАТУРА

1. Мобильная газета бизнеса [Электронный ресурс]. – Режим доступа: https://mobile.business-gazeta.ru/article/346729. – Дата доступа: 19.09.2017.
2. METANIT.COM.NodeJS и MongoDB [Электронный ресурс]. – Режим доступа: https://metanit.com/web/nodejs/6.1.php. – Дата доступа: 23.09.2015.
3. Express [Электронный ресурс]. – Режим доступа: http://expressjs.com/ru/. – Дата доступа: 25.09.2015.
4. METANIT.COM.Руководство по Angular [Электронный ресурс]. – Режим доступа: https://metanit.com/web/angular2. – Дата доступа: 25.09.2017.