DEVELOPMENT OF AN ELECTRONIC REAL ESTATE PLATFORM

Assoc. Prof. PhD Dimitar Minchev, Burgas Free University, mitko@bfu.bg
Zornitsa Nikolova, graduate student, zori_nikolova97@abv.bg

Abstract: This publication presents the development of an electronic real estate platform, implemented through the use of the following technologies: JavaScript, ReactJs. NodeJs, MongoDb.

Keywords: JavaScript, NodeJs, ReactJs, MongoDb.

РАЗРАБОТКА НА ЕЛЕКТРОННА ПЛАТФОРМА ЗА НЕДВИЖИМИ ИМОТИ

доц. д-р Димитър Минчев, Бургаски свободен университет, mitko@bfu.bg
Зорница Николова, дипломант, zori_nikolova97@abv.bg

Абстракт: Тази публикация представя разработка на електронна платформа за недвижими имоти, реализиране посредством използването на технологиите: JavaScript, ReactJs. NodeJs, MongoDb.

Ключови думи: JavaScript, NodeJs, ReactJs, MongoDb.

Увод

През последните няколко години се наблюдава бурно и динамично развитие на пазара на недвижими имоти в цялата страна. Затвърждава се тенденцията, че имотите стават все по-достъпни, а активността както при запитванията, така и при сключените сделки се е увеличава с всеки изминал ден. Справка в агенцията по вписванията за първите девет месеца на 2022 година показва, че с недвижимо имущество в България, включващо 113 общини са направени общо 175262 имотни сделки за продажба и 23774 сделки за наем на недвижимо имущество [1]. Макар и тенденцията след третото тримесечие на годината да е нормализиране на пазара (за това свидетелства отчета на Агенцията по вписванията, която за първи път от 15 години насам отчита спад в сделките с недвижимост през третото тримесечие), то все още може да се твърди, че жилищният пазар у нас е в подем. В своята същност, осъществяването на сделки на пазара за недвижими имоти представлява сложен и продължителен процес, които от своя страна изисква определени и специфични познания на участниците в него. В повечето от сключените сделки, субектите (продавач/наемодател и купувач/наемател) биват представлявани от

определен посредник. В качеството си на посредник на пазара на недвижими имоти се явява най-често агенцията за недвижими имоти.
Успехът на всяка една агенция занимаваща се с недвижимо имущество зависи от редица фактори, често разглеждащи се неразделно един от друг, а именно: - квалификацията, уменията и ценза на брокерите на недвижими имущества, доверието, което фирмата е създала и утвърдила у своите клиенти, както и коректното отношение към тях, управлението на предоставените им от клиентите ресурси, както и достъпността и визията на фирмата за самите клиенти. Докато, самия подбор на служители, тяхната мотивация и обучение в една такава агенция зависи изцяло и е приоритетна функция на управленския ѝ състав, понякога в стремежа си да подготвят най-добрите кадри се пренебрегва необходимостта фирмата да бъде лесно разпознаваема от всички бъдещи и настоящи клиенти. Развитието на съвременните дигитални и мрежови технологии доведоха до бум в предлагането на различни услуги и стоки, които са широко достъпни през глобалната интернет мрежа. Днес е немислимо да имаш успешен бизнес, без да имаш подходящо представяне в уеб пространството. В природата на хората е, първо да си направят собствено проучване на това което ги интересува и чак тогава да се обърнат към специалист, който да ги ориентира и/или им помогне в случаите, че не са доволни от постигнати собствени резултати. В много случаи, хората са привлечени от визията по която дадена компания е представена в дигиталния свят. Когато се говори за визия, тук трябва да се обърне внимание не само на това колко атрактивен е даден фирмен сайт/портал, но и колко лесно и интуитивно човек намира необходимата информация в него.
Целта на тази публикация е да покаже отделните етапи при създаването на специализиран сайт за използване от фирма/агенция занимаваща се с продажба, покупка, отдаване и наемане на различни видове недвижими имущества, като едновременно с това се използват съвременните реактивни JS технологии.

Обзор на съществуващи решения в областта

При създаването на всеки един уеб сайт е необходимо, разработчика да има изградена ясна концепция за разработката, която му предстои да реализира. За да се избере най-правилното и осведомено решение е необходимо да се анализира голям обем от информация, която в днешно време е достъпна до всеки само с един „клик“.

Каква е целта на сайта?

Основната цел на разработения сайт е да дава информация, която да улеснява всеки потребител при намирането ѝ, независимо от възраст, опит и умения за работа в интернет. Идеята е, информацията да бъде поднесена във вид, разбираем за всеки и едновременно с това да бъде изчерпателна. Освен информативната част, сайта трябва да може да предлага сигурност на всеки регистриран потребител от гледна точка на защита на личните му данни, кога и колко пъти е посещавал сайта, какви сделки е инициирал, какви имоти е обявил и т.н.

Целева група

При разработването на конкретния сайт, не са поставени конкретни изисквания към целевата група, която трябва да има достъп до него.

Съдържание на сайта

В своята същност сайта за недвижими имоти ще предлага на клиенти възможност за предварително запознаване по дефинирани от тях критерии, за пазарното наличие на недвижими имоти. Какво се визира под предварително запознаване? Тук основното съдържание, което ще бъде предоставено на посетителя на сайта е под формата на визуална информация – снимков материал, който трябва едновременно да бъде информативен, а от друга страна да бъде щадящ ресурсите на сървърната система, върху която ще се зарежда сайта. Сайта трябва да позволява на желаещите потребители да си създават собствени профилни акаунти, да си дефинират ролята – наемател/наемодател. Да инициират сделки и т.н.

Резултат от създаването на сайта

От икономическа гледна точка, тук поставената цел е основно да се увеличи не само посещаемостта на конкретния сайт, но и да доведе до повишаване реномето – доверието в съответната агенция за недвижими имоти, както и повишаване на реализираните имотни сделки, водещи до повишаване на съответните печалби за компанията, т.е. сайта ще има подчертано комерсиална цел.

За да може да се отговори в конкретика на всички тези въпроси е необходимо да се проучи текущото състояние на подобни уеб сайтове, което е направено в следващите под точки.

Обзор на вече съществуващи решения

За да може да се направи обстоен и пълноценен анализ на различните и достъпни в интернет пространство сайтове, първоначално е необходимо да се дефинират, конкретните параметри по които ще бъдат сравнявани намерените сайтове.

Самото обхождане на всички сайтове, практически е невъзможно – търсене в най-популярната търсачка Google на фразата „агенция за недвижими имоти“ показа наличието на над милион сайта, които в своето описание съдържат този израз. При търсене с англоезичния термин „real estate agency”, показаните резултати са милиард и осемстотин милиона.

Анализ на съществуващи фирмени сайтове

При анализа на съществуващи фирмени сайтове ще се акцентира върху следните няколко аспекта: Общ изглед на сайта при първоначално зареждане; Функционалност на сайта и организация на предоставяната информация; Допълнителни възможности; Използвани web технологии. Подборът на конкретно анализираните сайтове е на случаен принцип.

Номер	Линк	Технологии
1	https://yavlena.com	Microsoft IIS, ASP.NET, jQuery, Bootstrap
2	https://imoti.bg	LAMP, PHP, jQuery, Bootstrap
3	https://realistimo.com/bg/	LAMP, PHP, JavaScript, HTML, CSS

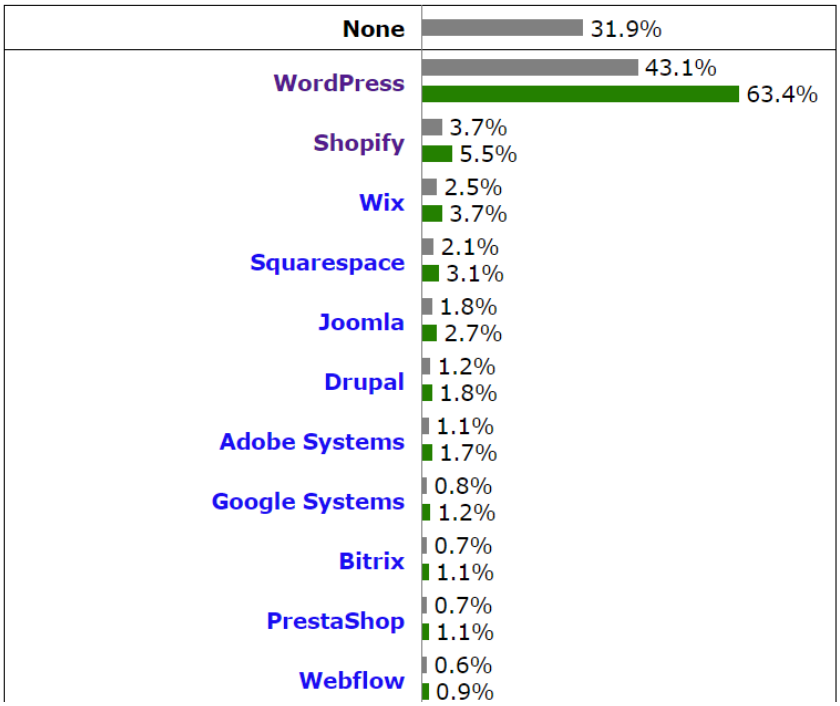
Показано бе какви са съвременните тенденции при изграждането на сайт за промотиране на недвижими имоти от гледна точка на визуалното представяне на информацията и какви функционалности е добре да съчетава в себе си съответния сайт.

Най-често използваните технологии при изграждане на web сайтове.

Системи за управление на съдържанието

Система за управление на съдържанието (Content Management System – CMS) е софтуер, който помага на потребителите да създават, управляват и променят съдържанието на един web сайт без да имат необходимостта от специализирани технически познания. Или по-общо казано, CMS е инструмент, който ще помогне на потребителя да създаде web сайт без да му се налага да го изгражда от нула или дори без да му се налага да програмира въобще.

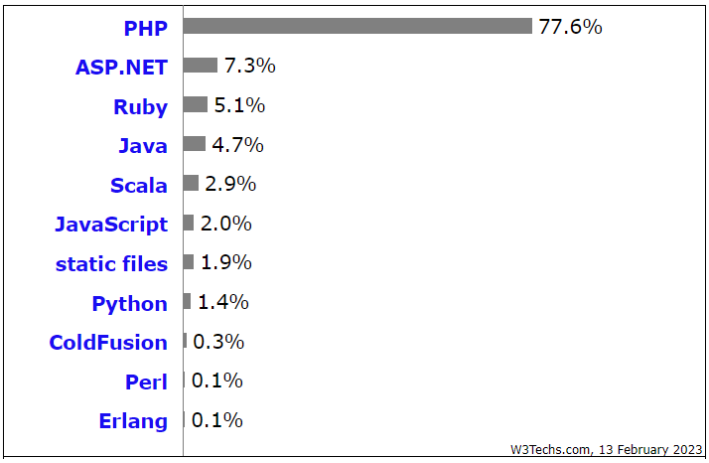
На Фиг. 0. е представена информация за най-често използваните системи за управление на съдържанието на web страниците, взета от (https://w3techs.com/). Тъй като списъка на системите, които се наблюдават от сайта наброява 880, то на фигурата са показани само тези, които са близо до и над 1 % използваемост. Тук трябва да се посочи и, че 31.9% от анализираните от w3techs сайтове не използват нито една от анализираните от тях системи за управление на съдържанието, а най голям дял се пада на WordPress с цели 63.4% или 43.1% за всички анализирани web сайтове, които използват система за управление на съдържанието.



Фиг. 0. Най-често използваните системи за управление на съдържанието.

Програмни езици използвани за сървърната част на сайта

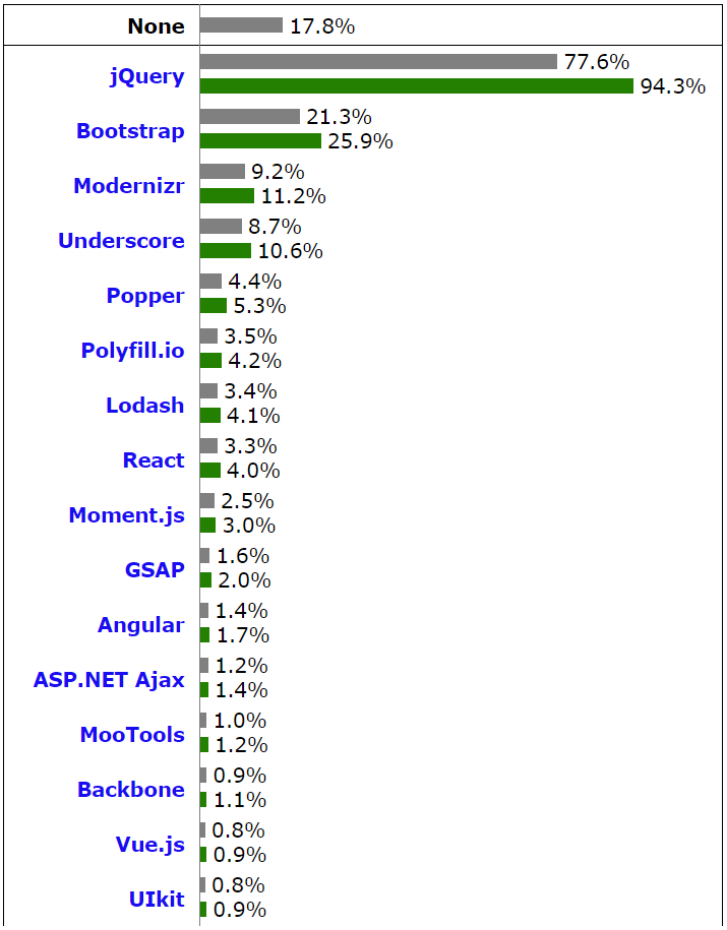
Следващия показател, който представлява интерес за разработчика на web приложения и сайтове, е какъв програмен език най-често се използва от страна на сървърното приложение. На Фиг.. са показани сървърните езици, които са се утвърдили при написването на приложенията, които си взаимодействат „недивидимо“ с потребителския интерфейс. Както може да се види от фигурата, отново с голяма преднина в сравнение с останалите е скриптовия език PHP с употреба от 77.6% за всички анализирани сайтове от w3techs.com. На втора позиция е ASP.NET на Макрософт с 7.3%, следван от Ruby – 5.1% и Java с 4.7%.



Фиг..2 Най-често използваните програмни езици за реализиране на сървърната част на web приложения и страници.

1.7.3. Най-често използвани библиотеки (JavaScript)

На Фиг. са показани най-утвърдените библиотеки, които се използват в днешно време от разработчиците на web приложения за изграждане на потребителския интерфейс на една фирмена страница.

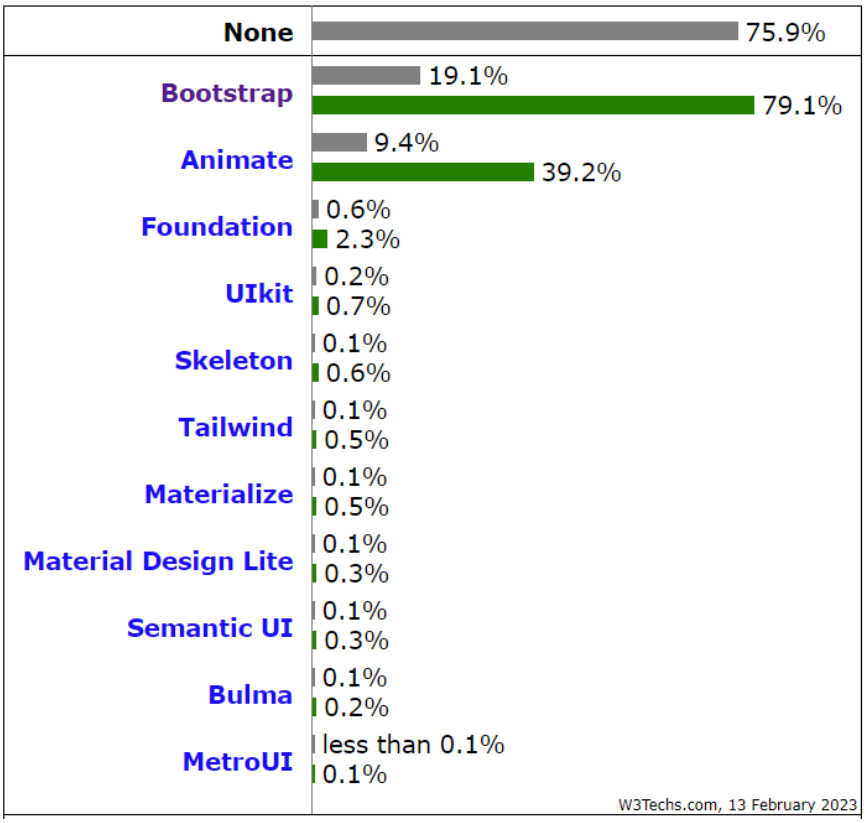


Фиг. 3 Дял на най-често използваните библиотеки за JavaScript

Както може да се види от фигурата, JQuery е най-популярната библиотека използвана от разработчиците с 77.6% следвана от Bootstrap с 21.3%, а третото място се пада на Modernizr с 9.2%. употреба над 5% има само Underscore библиотеката, а 9 библиотеки си разпределят местата между 1 и 5%.

Фреймуърк за каскадни стилове

От показаните на Фиг. използвани фреймуърк за създаване на каскадни стилове за web страници, може да се заключи, че в по-голямата си част, разработчиците, предпочитат да не използват такива – 75.9 от анализираните сайтове, не използват конкретен фреймуърк.

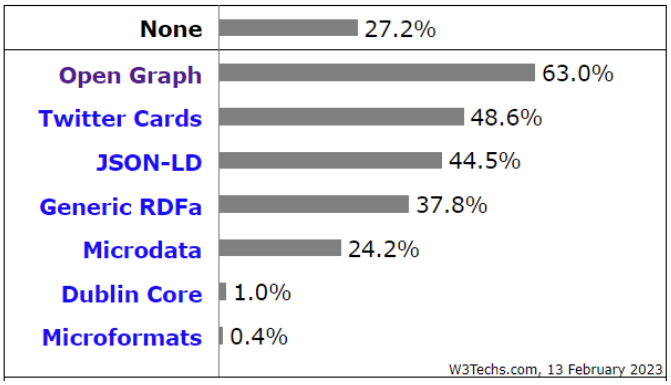


Фиг. 4 Най-често използваните фреймуърк за създаване на каскадни CSS стилове.

На първо място се нарежда Bootstrap фреймуърк-а с 19.1%, следван от Animate с 9.4% и всички останали имат дялово участие под 1%.

Технологии за структуриране на данните

На Фиг. са показани най-често използваните технологии/протоколи за структуриране на данните. Тук лидерско място заема протоколът, разработен първоначално от Facebook – Open Graph с 63 процента.



Фиг. 5. Технологии/протоколи за структуриране на данните

Избор и обосновка на използвани технологии

Тук ще бъдат разгледани набор от съществуващите съвременни технологии, които се използват при реализацията на уеб базирани комерсиални платформи.

PHP

PHP е сървърен скрипт език, проектиран специално за web нужди. PHP кодът може да се вгради в HTML страница и се изпълнява при всяко нейно посещение. Кодът се интерпретира от Web сървъра и генерира HTML или други изходни данни, които посетителя на сайта вижда. Чрез изпълняване на PHP програми на сървъра, може да се създават много мощни приложения, които си взаимодействат с базата данни и динамично генерират съдържание.

HTML и CSS

Езикът HTML (Hyper Text Mark-up Language) е универсалният език за създаване на webстраници, с чиято помощ може да се форматира текст, да се добавят графики, звук. Той е прост механизъм с вградени функции за форматиране и свързване на документи, за работа с таблици, фонови изображения и др. Страниците разработени с помощта на HTML като цяло са статични. Едно от най-големите преимущества на HTML е неговата лесна реализация. Употребата на каскадни стилове в разработката на един web сайт е голямо улеснение и удобство за неговият проектант. С тяхна помощ може лесно да се контролира и уеднакви вида и поведението на различните елементи в създадената HTML страница.

EcmaScipt 6

В своята същност, езикът JavaScript и неговите характеристики са дефинирани в стандартът ECMA-262. Това води до по общо възприето в програмните среди този език да се дефинира и да се нарична ECMAScript. Когато се говори за JavaScript по отношение на Node.js или във връзка с използванетона интернет браузър, се визира именно разширението на ECMAScript. Браузърите и Node.js от своя страна добавят допълнителна функционалност на скриптачрез добавянето на допълнителни обекти и методи, но в своята цялост ядрото на езика е това което е първоначално дефинирано в ECMAScript. Продължаващото разработване на ECMA-262 е основополагащо за успехът на JavaScript като цяло, а ECMAScript 6 е водещата и най-актуална версия на езика, като през месец Юни 2022 година излезе и последната 13-та актуализация на спецификациите на стандарта.

React

Към днешна дата, съществуват множество JavaScript MVC (Model View Controller) библиотеки. Първоначално React е създаден от разработчиците на Facebook. React е библиотека са създаване на композитни, потребителски интерфейси. React наистина е мощен инструмент, когато става въпрос за динамичното обновяване на данните.

Material Design и Material UI

Material Design е дизайнерски език, предсавем за първи път от Google 2014 г. В своята същност, това е возуален език, който използва решетъчно оформление на дизайна, реагиращи анимации, визуални компоненти задаващи дълбочина, осветеност и сенки на компонентите в приложението. Целта на Material Design се свежда до три основни неща: създаване, обединявне и персонализиране.

Node.js

Node.js е сървърна платформа за изпълнение на JavaScript код. Node.js (или само Node) е сървърно приложение, създаващо среда за изпълняване на JavaScript код (runtime environment), а не език за програмиране. Node.js прави възможно изпълнението на JavaScript код директно на сървъра, както обикновено това се случва в уеб браузъра.

Visual Studio Code

Visual Studio Code комбинира простотата на обиновенните редактори на програмен код с мощни инструменти за разработка като IntelliSense автоматично довършване при писане, проследяване и отстраняване на грешки. Редакторът поддържа macOS, Linux, and Windows - за да може да бъде максимално достъпен без операционната система да е пречка.

MongoDB

MongoDB е система за управление на бази от данни (СУБД), създадена за бърза разработка на уеб приложения, и по тази причина е проектирана да дава по-голяма свобода на програмистите при проектиране на базата от данни.

Разработка на платформа за недвижими имоти

За постигането на проектните изисквания и успешната реализация на сайта, за потребителския интерфейс ще се използва скриптовия език JavaScript, React фреймуърк, ReactRouter за навигация, Formik+Yup за валидация Redux Persistent за локално съхранение на данните и React drop zone за ъплоуд на изображения. За сървърната част, ще се използват NodeJS и ExpressJS, като среда в която да се зарежда web приложението, Mongoose за менажиране на MongoDB и Multer за съхранение на изображенията. Като развойна среда ще се използва Visual Studio Code на Microsoft.

Първоначални настройки на развойната среда и инициализация на проекта.

За успешната реализация на сайта, практическото му оформление ще бъде разделено на две части: сървърна част, която ще функционира невидимо за потребителите и клиентска част, която ще си взаимодейства с потребителите. За разработката на сайта ще се използва средата на Microsoft – Visual Studio Code, в която на практика ще се създадат два отделни, но взаимно свързани проекта, условно наречени server и client. Приложението, което ще се създаде ще е React базирано и за да функционира сигурно, надеждно и безопасно от гледна точка на съхранение на данните на потребителите и на

самата агенция, първоначално ще бъде показана разработката на сървърната част, в която ще се настроят всички компоненти свързани с автентикацията, регистрацията, нулиране на пароли и създаването на базата данни в която потребителите, да могат да записват невидимо за тях своите оферти.

Реализация на сървърната част на приложението.

За правилното функциониране на сървърното приложение и да може то автоматично да се обновява при записване на промени в кода, първоначално е необходимо да се инсталира nodemon, както и е необходимо да бъдат инсталирани следните пакети: express; morgan; mongoos; jsonwebtoken; bcrypt; nanoid; cors; email-validator; slugify; node-geocoder. npm инсталаторът подържа възможността всички пакети да бъдат инсталирани с една единствена команда/

Първият пакет който се инсталира е express. Философията на Express е да предостави малки и стабилни инструменти за HTTP сървъри, което го прави чудесно решение за приложения на уебсайтове изградени на принципа single page application.

За криптирането на паролите на отделните потребители се инсталира библиотеката bcrypt. Самата bcrypt функция превръща потребителската парола в низ, който може да бъде с големина до 72 байта и на практика, прави декриптирането на паролата невъзможно, за кратък интервал от време, с което се повишава сигурността на чувствителните данни.

CORS или Cross-Origin Resource Sharing, е механизъм за споделяне на ресурси с кръстосан произход по отношение на протокол, име на хост или порт, като за целта се използва HTTP-header частта в браузъра. По този начин, дори и неизвестен по произход, заявителя получава разрешение за достъп и зареждане на ресурси. Пакетът cors, наличен в npm, се използва за справяне с CORS грешки в едно типично Node.js приложение.

morgan и jsonwebtoken се използват съответно за подсигуряване на HTTP заявките, за създаването на лог с HTTP заявките и съответните грешки и респективно за автентификация на достъпа.

За работа с базата данни MongoDB е необходимо инсталирането на mongoose, а за имаме възможноста да генерираме уникални ID-та на потребителите ще използваме nanoid.

email-validator, както самото име подсказва, ще се използва за валидиране на въвежданите имейл адреси.

slugify ще се използва за задаване на четимо за потребителя в адресната лента на браузъра име.

Създаване на основния сървърен и конфигурационен файл

След инициализация на проекта е необходимо да се създаде NodeJS сървъра. За целта в папката server създаваме файл – server.js.

За връзка между express и междинния (middleware) софтуер, създаваме една променлива app, инициализираме q, а указването, че ще се използва определен помощен софтуер става, чрез функцията use. Стартирането на сървъра може да стане чрез командите npm start или nodemon server.js.

Всички конфигурации необходими за нормалната работа на сървърното приложение, ще бъдат дефинирани в отделен конфигурационен файл, наречен config.js.

По-долу е показан отрязък от кода в config.js файла. Както може да се види в него се записват всички ключове за достъп до различните глобално достъпни платформи които ще се използват за създаването на необходимата функционалност на сайта:

```
import SES from "aws-sdk/clients/ses.js";
```

```
//upload image
import S3 from "aws-sdk/clients/s3.js";
// get geo location using google places API
import NodeGeocoder from "node-geocoder";

export const DATABASE = "mongodb+srv://Zaya:**********@lagoon-
re.*******.mongodb.net/?retryWrites=true&w=majority";
// export const DATABASE = "mongodb://127.0.0.1:27017/lagoon-real-estate";

export const AWS_ACCESS_KEY_ID = "********************";
export const AWS_SECRET_ACCESS_KEY = "****************************************";

export const EMAIL_FROM = '"Lagoon RE" <zori_nikolova97@abv.bg>';
export const REPLY_TO = "zori_nikolova97@abv.bg";

const awsConfig = {
   accessKeyId: AWS_ACCESS_KEY_ID,
   secretAccessKey: AWS_SECRET_ACCESS_KEY,
   region: 'eu-central-1',
   apiVersion: '2010-12-01',
};
export const AWSSES = new SES(awsConfig);
// for uploading image
export const AWSS3 = new S3(awsConfig);

const options = {
   provider: "google",
   apiKey: "****************************************",
   formatter: null,
};

export const GOOGLE_GEOCODER = NodeGeocoder(options);

export const JWT_SECRET = '*************************';
export const CLIENT_URL = 'http://localhost:3000';
```

За защита на личната и чувствителната информация, ключовете за достъп до базата с данни и приложния интерфейс към платформата Google Places са премахнати.

Използване на MongoDB Atlas за създаване на базата с данни

В случай, че няма възможност за локално инсталиране на MongoDB, то MongoDB Atlas (https://www.mongodb.com/) е web базирана платформа, която предоставя различни опции за използване на базата с данни, като за тестови нужди със съответните ограничения, базата може да се ползва безплатно. За да може да се използва е необходимо да се създаде потребителски акаунт, от който в последствие да се генерират необходимите ключове за достъп.

Веднъж създадена базата е необходимо да се направят необходимите промени съответно в server.js и config.js файловете така, че базата да бъде достъпна.

По-долу е показан кодът, необходим да се добави в server.js файла:

```
import mongoose from "mongoose";
import { DATABASE } from "./config.js";
// db
mongoose.set("strictQuery", false);
mongoose
  .connect(DATABASE)
  .then(() => console.log("db_connected"))
  .catch((err) => console.log(err));
```

За да имаме напълно функционална база с данни, то е необходимо в нея да има съответните записи. За целта е необходимо да се създаде модел на съответните записи. Следващия отрязък от код демонстрира създаването на модели за потребител и имот, които ще бъдат записвани в MongoDB.

```
import { model, Schema, ObjectId } from "mongoose";
const schema = new Schema(
  {
    username: {
      type: String,
      trim: true,
      required: true,
      unique: true,
      lowercase: true,
    },
    name: {
      type: String,
      trim: true,
      default: "",
    },
    email: {
      type: String,
      trim: true,
      required: true,
      unique: true,
      lowercase: true,
    },
    password: {
      type: String,
      required: true,
      maxLength: 256,
    },
    address: { type: String, default: "" },
    company: { type: String, default: "" },
    phone: { type: String, default: "" },
    photo: {},
    role: {
      type: [String],
      default: ["Buyer"],
      enum: ["Buyer", "Seller", "Admin"],
    },
    enquiredProperties: [{ type: ObjectId, ref: "Ad" }],
    wishlist: [{ type: ObjectId, ref: "Ad" }],
    resetCode: "",
  },
  { timestamps: true }
);
export default model("User", schema);
```

а/ модел на потребител

```
import { model, Schema, ObjectId } from "mongoose";
const schema = new Schema(
  {
    photos: [{}],
    price: { type: Number, maxLength: 255 },
    address: { type: String, maxLength: 255, required: true },
    bedrooms: Number,
    bathrooms: Number,
    landsize: String,
    carpark: Number,
    location: {
      type: {
        type: String,
        enum: ["Point"],
        default: "Point",
      },
      coordinates: {
        type: [Number],
        default: [23.280729, 42.725189],
      },
    },
    title: {
      type: String,
      maxLength: 255,
    },
    slug: {
      type: String,
      lowercase: true,
      unique: true,
    },
    description: {},
    postedBy: { type: ObjectId, ref: "User" },
    sold: { type: Boolean, default: false },
    googleMap: {},
    type: {
      type: String,
      default: "Other",
    },
    action: {
      type: String,
      default: "Sell",
    },
    views: {
      type: Number,
      default: 0,
    },
  },
  { timestamps: true }
);
export default model("Ad", schema);
```

б/ модел на имот

Както се вижда от фигурата в схемата на моделите се дефинират основните признаци и полета, които трябва да се попълнят от потребителя, когато се регистрира в сайта или когато иска да обяви за продажба или отдаване под наем на някакъв имот. Част от полетата са задължителни, required: true, докато има и такива, които не са, например полето name.

Маршрутизация на постъпващите заявки

Едно от съществените неща при изграждането на сървърното приложение, е то да може коректно да обработва постъпилите от клиентското приложение JSON заявки или така наречените “routes”. Всички тези заявки в последствие се пренасочват към съответните контроли, които в зависимост от подадените параметри или връщат съответния резултат или информират потребителя за възникнала грешка.

Добрата и утвърдена практика при изграждането на backend приложенията е отделните функционалности да бъдат групирани/обединени под общо наименование. В тази връзка, тук условно са обособени две групи маршрутизация: - за идентификация/автентикация на потребителите (./server/routes/auth.js) и за постъпилите обяви в сайта (./server/routes/ad.js).

За автентикация на всеки един потребител се използва jasonwebtoken или накратко JWT, като тази функционалност предоставя възможност след изтичане на определен период от време, потребителя отново да потвърди своята идентичност. На **Грешка! Източникът на препратката не е намерен.** е показан кода за декодиране на json web token. Всеки потребител след регистрация в сайта трябва да потвърди/валидира имейла си, с което сървъра ще му позволи да разглежда съдържанието на тези страници на сайта, за които не е нужно идентификация на потребителите.

Криптиране на потребителските пароли и взаимодействие с потребителите

За сигурността на потребителските данни преди даден потребител да бъде съхранен в базата с данни, се използва bcrypt пакета. bcrypt представлява сам по себе си криптографски алгоритъм с възможност за хеширане на пароли.

За повишаване на сигурността от неоторизиран достъп е необходимо да се разпишат определени правила, които ако не бъдат спазени, потребителите да биват ограничавани и да не могат да извършват конкретни действия в сайта. Също така, сайтът трбява да предоставя възможност на потребителите, освен да се регистрират в него, да се логват, да ресетват своите пароли или да укажат, че са забравили своята парола. Следващите няколко абзаца с код демонстрират тази функционалност. Тъй като кода е доста обемист като съдържание, тук ще бъдат демонстрирани само тези имплементации, които са от съществено значение за сигурността на сайта.

Регистриране на потребителя в сайта

За да може да създава обяви, всеки потребител на сайта трябва първоначално да се регистрира в него. След успешна регистрация и с посочен валиден e-mail адрес на този адрес, ще му бъде изпратен линк с които да удостовери, че действително такъв потребител съществува.

Освен изпращане на линк за потвърждение от страна на потребителя, е предвидено допълнително и проверка за въведен валиден адрес, т.е. ако липсва символа “@” за принадлежност към конкретен домейн, проверка за въведена парола, проверка за броя на символите в паролата и ако е под един определен минимум, то тя се възприема като невалидна, както и проверка, дали вече няма потребители, които са използвали

посочената парола. Демонстрация на тези функционалности и проверки, ще бъдат показани в глава 4 на дипломната работа.

Логване на потребителя в сайта

За тази цел, потребителя трбява да въведе паролата с която се е регистрирал и ако след сравнение на хешираната парола, съхранена в базата ни с данни, то тя съответства на въведената, то потребителя ще може успешно да влезе в сайта.

Достъп до акаунта при забравена парола

В случаите на забравена парола, потребителят трябва да има възможност да влезе в своя профил, като за цетлта на посочения имейл, ще му бъде изпратен код, който да потвърди и който автоматично да му позволи да влезе в профила си.

Първоначално се проверява дали съществува потребител с въведения email и ако такъв не съществува, то се извежда съобщение за грешка. В случаите, че потребителят, забравил паролата си, вече е регистриран в сайта и има запис за него в базата с данни, то тогава неговото ID ще бъде ресетнато, като в същото време ще му бъде изпратен на посочения и валиден email линк със който да може да достъпи своя акаунт.

Реализация на клиентската/потребителската част на приложението

За първоначална инициализация и инсталиране на необходимите пакети и модули на потребителската част на приложението е необходимо да се изпълни команда: npx create-react-app <име на проекта>. В случаите, че предварително е избрана директорията, то името на проекта не е задължително да се посочва, а вместо него командата придобива следния вид: npx create-react-app ./

Първоначалната идея бе, визуалните компоненти на сайта да бъдат изградени на базата на MaterialUI, но в последствие, бе избрана да се ползва библиотеката на Bootstrap (https://getbootstrap.com/), като за конкретната реализация се използват свободно достъпните теми от Bootswatch (https://bootswatch.com/).Тази библиотека разполага с богата гама от компоненти, форми, теми и всевъзможни инструменти, което значително улеснява разработчика.

За да може да се използва библиотеката, в генерираната проектна директория public, във файла index.html, е необходимо да се добави линк към съответния избран стил.

В случай, че в последствие потребителя, желае да промени облика на вече създадения сайт, то е достатъчно само да намери отговаряща на неговите изисквания тема и да укаже новия линк във файла.

Маршрутизация в потребителските страници

Оснвната разлика между приложенията от тип приложение с една страница и web приложенията където страниците се зареждат в потребителя, е че цялата информация при SPA се зарежда заедно със всички необходими активи и всяко последващо отваряне на страницата или последващи страници не изисква допълнителни заявки до сървъра. Това налага при първоначалното зареждане, всички маршрути към по-следващите страници да бъдат предварително указни. За навигацията между отделните страници се използва react-router-dom пакета.

В случай, че търсената страница, не може да бъде намерена, на потребителя ще му бъде изведено съобщение за грешка. Рутирането между отделните страници става в основното приложение App.js, а по-долу е показана главната функция App() чрез която се реализира тази функционалност.

```
function App() {
  return (
```

```
    <BrowserRouter>
      <AuthProvider>
        <SearchProvider>
          <Main />
          <Toaster />
          <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/login" element={<Login />} />
            <Route path="/register" element={<Register />} />
            <Route
              path="/auth/account-activate/:token"
              element={<AccountActivate />}
            />
            <Route path="/auth/forgot-password" element={<ForgotPassword />} />
            <Route
              path="/auth/access-account/:token"
              element={<AccessAccount />}
            />

            <Route path="/" element={<PrivateRoute />}>
              <Route path="dashboard" element={<Dashboard />} />
              <Route path="ad/create" element={<AdCreate />} />
              <Route path="ad/create/sell/house" element={<SellHouse />} />
              <Route path="ad/create/sell/land" element={<SellLand />} />
              <Route path="ad/create/rent/house" element={<RentHouse />} />
              <Route path="ad/create/rent/land" element={<RentLand />} />
              <Route path="user/profile" element={<Profile />} />
              <Route path="user/settings" element={<Settings />} />
              <Route path="user/ad/:slug" element={<AdEdit />} />
              <Route path="user/wishlist" element={<Wishlist />} />
              <Route path="user/enquiries" element={<Enquiries />} />
            </Route>

            <Route path="/ad/:slug" element={<AdView />} />
            <Route path="/agents" element={<Agents />} />
            <Route path="/agent/:username" element={<Agent />} />

            <Route path="/buy" element={<Buy />} />
            <Route path="/rent" element={<Rent />} />
            <Route path="/search" element={<Search />} />
            <Route path="*" element={<PageNotFound />} />
          </Routes>
          <Footer />
        </SearchProvider>
      </AuthProvider>
    </BrowserRouter>
  );
}
```

Създаване на навигационно и потребителско меню

За да може потребителя да навигира между отделните подсайтове, е необходимо да се създаде меню за навигация.

Главното меню е оформено в две части. Първата част е лентата с препратки към отделните страници, а втората част на менюто показва дали потребителя е вляал в сайта или не. В случай, че потребителят е анонимен, в тази част се изписва <Register>, с което се подканя потребителя да направи регистрация в сайта. Когато потребителят е влязъл в сайта, то в това поле се изписва потребителското му име с което се е регистрирал.

Потребителското меню за разлика от навигационното е оформено като неподреден списък. От това меню се отварят различните потребителски форми, които включват: Потребителска дъска с обяви – Dashboard; Дъска с желания на потребителя – Wishlist;

Запитвания от потребителя – Enquiries; Създаване на обява – Cread Ad; Профил на потребителя – Profile; Настройки – Settings;

Използване на Google places и Google map за визуализиране на локацията на имота

Google Places се използва за подпомагане на потребителя при въвеждане на желана локация. Тази функционалност му показва възможните дестинации в зависимост от броя на въведените символи. Използването на Google Places изисква получаването на API ключ, който се използва за получаване на достъп до този вид услуги.

Другата функционалност, която е използвана при създаването на сайта е Google Map. Чрез нея всеки потребител, ще може да види, точното географско положение на търсения от него имот.

Поради големият обем на прогамният код – повече от 2800 реда, е невъзможно всички реализирани функционалности да бъдат описани в детайли. Тук бяха представени тези, които да дадат обща представа за възможностите на практическата реализация, а по-подробна информация за нея, като възможности ще бъдат представени в глава 4 на настоящата дипломна работа

Изследване и анализ на разработката

В тази глава ще бъдат показани основните показани отделни етапи свързани с оживяването на сайта. Необходимо е да се направи предварителното уточнение, че голяма част от фигурите показани в следващите няколко абзаца, са правени по време на самото написване на програмния код и в различни времеви периоди, тъй като изграждането на един напълно завършен сайт изисква дълбоко вникване и запознаване с утвърдилите се технологии, като в много от случаите това е довело до преминаване от един вид библиотека към друг с цел покриване нуждите на проектното задание.

И в тази глава последователноста на изследването ще следва концепцията на изграждането на сайта. Първо ще бъдат показани тези моменти имащи отношение към сървърната част на приложението, а след това и към клиентската част.

Изследване функционалността на NodeJS сървъра

Като помощно средство при настройване на заявките към и от сървъра е използван програмният продукт Postman (https://www.postman.com/). Това приложение дава възможност към сървъра да се изпращат различни типове заявки в JSON формат и да се визуализира върнатият от сървъра отговор.

Изследване отговора на сървъра при регистриране на нов потребител

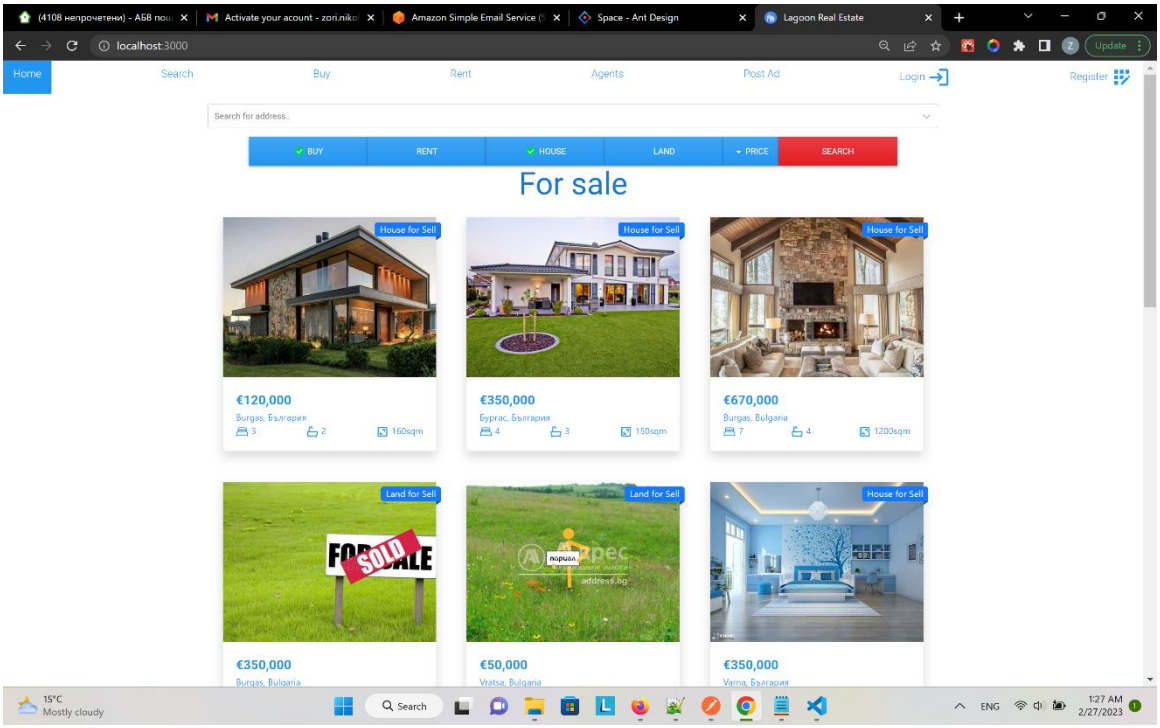
При регистриране на нов потребител, сървъра изпраща обратен email към потребителя, който трябва да потвърди, че той е действителния собственик на посочения от него email, чрез щракването на генерирания от сървъра линк за потвърждение. В случай, че потребителя, закъснее с повече от час, да отвори изпратения му линк, то генерирания токън става с което регистрацията на потребителя се отхвърля. В такъв случай, потребителя трябва да поиска подновяване на токъна за да приключи с успешно с регистрацията в сайта. Веднъж вече, успешно регистрирал се потребител, чийто email е бил валидиран, не може да бъде регистриран повторно със същия email . В този случай сървъра ще върне отговор, че email-а вече е зает, а потребителят ще трябва да въведе нов. Аналогично при опит за регистрация в сайта без да бъде посочена парола, сървъра ще отхвърли регистрацията, като изведе съобщение с което да информира потребителя, че паролата е задължителна.

Изследване записите на обявите в базата с данни

Всяка обява в сайта се записва в базата с дани след публикуването и на сайта. При добавяне на една примерна обява **Грешка! Източникът на препратката не е намерен.**сървъра генерира JSON пакет за осъществяване на примерен запис. От всички направени тестове на функционалността на сървърното приложение, то функционира спрямо очакванията и в съответствие с програмния код.

Изследване общия облик на потребителския сайт

На Фиг. е поаказана началната страница **Home** при зареждане на приложението. От фигурата се вижда, че всички налични обяви могат да бъдат разгледани дори и от нерегистрирани потребители.



Фиг. 6 Начална страница на потребителското приложение

Както може да се види от направените изследвания, и потребителската и сървърната част функционират в синхрон и според зададените програмни параметри и код.

Заключение и перспективи за развитие

Разработената уеб платформа за агенция за недвижими имоти съчетава предимствата на най-новите и съвременни технологии за изграждането на едностранични web приложения, които все по-често са предпочитани в сравнение с уеб приложенията, които е необходимо да изпращат заявки до сървърните приложения при зареждане на нова страница в браузъра. Това от своя страна поставя пред разработчика/разработчиците непрекъснато надграждане на своите знания, произлизащо от динамиката на IT технологиите. Самото проектиране на една такава платформа изисква значителни познания, както в областта на backend концепциите, така и съществени знания свързани със създаването на приложния интерфейс, чрез който всеки един потребител си взаимодейства с приложението. Именно такава концепция бе приета и при създаването на представената web платформа – сървърно приложение, което да работи невидимо за потребителите и което да следи всички заявки от страна на потребителите да се

изпълняват според имплементирания в него код и клиентско приложение, което от една страна да помага на потребителите и което да е в крак със съвременните тенденции.

Използвана литература и интернет източници

1. https://www.registryagency.bg/bg/registri/imoten-registar/statistika/
2. https://bootswatch.com/.
3. https://domm.bg/.
4. https://getbootstrap.com/.
5. https://imoti.bg/.
6. https://realistimo.com/bg/.
7. https://w3techs.com/.
8. https://www.mongodb.com/.
9. https://www.postman.com/.
10. https://www.yavlena.com/.