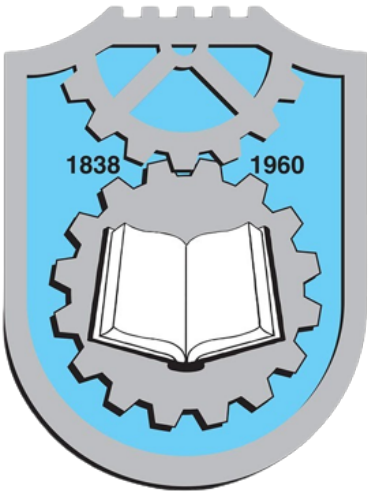
QA Analiza aplikacije *TravelApp*

Đorđe Karišić - C.L.A.Y.

Februar 2023.



Fakultet Inženjerskih nauka,
Univerzitet u Kragujevcu

Sadržaj

1 Uvod

U toku i nakon razvoja prethodno imenovane aplikacije javlja se potreba za analizom koda iste. Data aplikacija je podeljena na više delova po funkcionalnosti, koji su pakovani u klasne objekte i statičke klase čijim metodama se definiše određeno dešavanje/ponašanje. Kako bi se jasno razdvojili koncepti testiranja i efikasno pokrila logika implementacije, potrebno je grupisati testove u strukture različite po elementima koje testiraju.

- Linije koda
- Funkcije i metode
- Klase i osobine

Potpuna implementacija definisana je u programskom jeziku **PHP**, tako da će se kao alat za testiranje koristiti **PHPUnit** framework, čije mogućnosti zadovoljavaju potrebe testiranja, osim u slučaju *Mock*-ovanja i *Stub*-ovanja statičke klase.

PHPUnit pruža uslugu generisanja analize projekta u vidu web stranice, tekstualnog dokumenta i prikazivanja u terminal. Za svrhu analize ovog projekta koristiće se prikaz analize u vidu web stranice.

Pored Unit testing-a, potrebno je osigurati smisleno i precizno generisanje podataka. Za projektovanje sistema sposobnog za dati rad koristiće se Python sa sledećim pratećim bibliotekama:

Selenium Prikupljanje podataka sa web stranica.

Pandas Skladištenje i manipulacija podataka.

Numpy Operacije za nizove podataka (series,dataframe).

Faker Generisanje jednostavnih lažnih podataka.

OpenAI Generisanje složenijih lažnih podataka uz pomoć davinci-003 modela.

matplotlib Predstavljanje podataka graficima.

translators, bing image download, srtools Pripremanje podataka.

2 PHPUnit- Analiza izvornog koda

Analizom koda napisanog u okviru server-side razvoja aplikacije dolazi se do zaključka da se sve komponente sistema i prateće skripte služe metodama, instancama i poljima klasa definisanih u okviru foldera /**classes**. Folder /classes ima hijerarhiju prikazanu listom ispod.

- /classes
  - Config.php
  - Input.php
  - DB.php
  - Redirect.php
  - Session.php
  - Token.php
  - User.php
  - Validate.php

2.1 Config.php

```php
<?php

class Config
{
    public static function get($path)
    {
        $config = $GLOBALS['config'];
        $path = explode('/', $path);

        foreach ($path as $bit) {
            if (is_array($config) && array_key_exists($bit, $config)) {
                $config = $config[$bit];
            } else {
                return false;
            }
        }
        return $config;
    }
}

?>
```

Svrha komponente Config.php je čuvanje konfiguracionog fajla za sesije, ime DB, porta, korisničkog imena...Klasa Config sadrži jednu statičku funkciju koja vraća mixed value.

```php
<?php

declare(strict_types=1);

require_once("core/init.php");

use PHPUnit\Framework\TestCase;

class ConfigTest extends TestCase
{
    public function test_error_assure()
    {
        $this->assertFalse(Config::get(implode(str_split(random_bytes(30)))));
    }
}

?>
```

Proverava se slučaj kada se kao argument prosledi string nasumičnih brojeva međusobno povezanih znakom '/', za koje se zna da nisu ključevi u globalnom nizu CONFIG. Očekivani output je logičko False, što govori o tome da se funkcija prekida u slučaju nepoznatih ulaznih podataka.

Ovaj test nije preterano značajan, ali u slučaju da u budućnosti dođe do promene ove funkcije, osigurava da se funkcija i dalje ponaša isto u ovakvim slučajevima.

2.2 DB.php

Fajl DB.php sadrži klasu DB koja služi za rad sa bazom podataka. Inicijalizacija instance ove klase je predstavljena obrascem Singleton, koji garantuje postojanje samo jedne instance ove klase. Sadrži polja pdo, query, error, results, count Sadrži sledeće funkcije:

getInstance statička funkcija, vraća objekat klase DB (singleton).

query za argumente prime SQL i parametre za bind-ovanje. Vraća objekat koji je pozvao metodu.

delete

action Uproštena verzija query funkcije. Za prosleđenu DML komandu, ime tabele i uslov vraća trenutni objekat sa izmenjenim unustrašnjim stanjima.

get Ograničava upotrebu funkcije action tako što kao prvi parametar prosleđuje SELECT *

insert Za ime tabele i liste parova ime kolone, vrednost unosi red u bazu.

updateUser Nalik funkcijama iznad, vraća True ukoliko je operacija uspešna, False ukoliko nije.

results Vraća privatno polje results objekta koji je pozvao metodu.

first Vraća prvi red rezultata sadržanih u polju results.

error Ukoliko se javila greška u izvršavanju neke od funkcija, poziv ove funkcija vraća niz grešaka, u suprotnom, False.

count Vraća broj redova privatnog polja results.

```php
<?php

class DB
{
    private static $_instance = null;
    private $_pdo,
            $_query,
            $_error = null,
            $_results,
            $_count = 0;

    public function __construct()
    {
        try {
            $this->_pdo = new PDO('mysql:host=' . Config::get('mysql/host') . ';dbname=' . Co
            $this->_pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
        } catch (PDOException $e) {
            die($e->getMessage());
        }
    }

    public static function getInstance()
    {
        if (!isset(self::$_instance)) {
            self::$_instance = new DB();
        }
        return self::$_instance;
    }
    ........
```

Klasa DBTest sadrži test metode analogne ovoj klasi.

```php
<?php
declare(strict_types=1);

require_once("core/init.php");

use PHPUnit\Framework\TestCase;

class DBTest extends TestCase
{
    public function test_if_returns_singleton()
    {
        #given CLASS DB, when getInstance called, then check if singleton is working
        $db =  DB::getInstance();
        $clone =  DB::getInstance();

        $this->assertEquals($db, $clone);
    }
    ?>
```

Prvi test će imati za zadatak testiranje Singleton obrasca. Jednostavno, za dve promenljive pozvati funkciju getInstance i proveriti da li su jednake.

```php
<?php
public function test_if_error_cleared()
    {
        $db =  DB::getInstance();
        $this->assertFalse($db->error());
    }
?>
```

Drugi test proverava da li je nakon inicijalizacije DB prazna lista grešaka.

```
public function test_if_PDO_query_concrete()
    {
        $this->expectException('TypeError');
        $db = DB::getInstance();
        $sql = "SELECT * from aranzmani WHERE aran_id = ?";
        $db->query($sql, 'abc');
    }
<?php
?>
```

Treći test za traženi tip vrednosti parametra fields '<array>' daje '<String>' i očekuje TypeError.

```php
<?php
public function test_if_PDO_query_checks()
    {
        $this->expectException('PDOException');
        $db = DB::getInstance();
        $sql = "SELECT * from aranzman WHERE aran_id = ";
        $db->query($sql, ['f']);
    }
?>
```

Četvrti test proverava da li dolazi do izuzetka PDO ukoliko prosleđeni parametar SQL ima grešku.

```php
<?php
public static function provideFakeActions()
    {
        return [
            ["SELECT *","aranzmani",["aran_id","==","1"]],
            ["SELECT *","aranzmani",["aran_id","!=","jedan"]],
            ["SELECT *","aranzmani",["aran_id","-","41"]],
            ["SELECT *","aranzmani",["aran_id","=="]]
        ];
    }
    /**
    * @dataProvider provideFakeActions
    */
    public function test_if_action_foolproof($action, $table, $where)
    {
        $db = DB::getInstance();
        $check = $db->action($action, $table, $where);
        $this->assertEquals($check, false, "" . ($check));
    }
    public function test_if_insert_foolproof()
    {
        $db = DB::getInstance();
        $check = $db->insert("aranzmani", []);
        $this->assertEquals(false, $check);
    }
?>
```

Ovaj test proverava da li se metode insert i provide ponašaju očekivano pri radu sa neočekivanim parametrima.

```php
<?php
public function test_if_update_user()
    {
        $db = DB::getInstance();
        $table = "korisnik";
        $fields = ["tip" => "1"];
        $db->insert($table, $fields);
        $res = $db->query("SELECT korisnik_id,tip FROM `korisnik` WHERE korisnik_id in (SELE

        $id = $res->first()->korisnik_id;
        $db->updateUser("korisnik", $id, ["tip" => "2"]);

        $res = $db->query("SELECT korisnik_id,tip FROM `korisnik` WHERE korisnik_id in (SELE

        $idN = $res->first()->korisnik_id;
        $db->delete("korisnik", ["korisnik_id","=",$idN]);
        $this->assertEquals($id, $idN);
    }
?>
```

Poslednji test proverava da li se identifikator novog korisnika, nakon update-a, poklapa sa identifikatorom korisnika pronađenog kao najnovijeg. Očekivani rezultat je True.

2.3 HTMLTest.php

Cilj klase HTMLTest je da prođe kroz sve fajlove koji sadrže tagove karakteristične za HTML5 dokument, i proveri da li postoji neželjeni output ili greška u DOM strukturi.

```php
<?php

declare (strict_types = 1);

require_once ("core/init.php");

use PHPUnit\Framework\TestCase;
class HTMLTest extends TestCase{

protected $traceError = TRUE;

public function test_if_undesirable_output(){
    $dir = 'C:/wamp64/www/PISiBP-TravelApp/';
    $appDir = new RecursiveDirectoryIterator($dir);
```

```
    $iterator = new RecursiveIteratorIterator($appDir);
    $matches = new RegexIterator($iterator, '/^.+\.php$/i', RecursiveRegexIterator::GET_MATC
    $chck = True;
    $where='';
    $what='';
    foreach ($matches as $phpFile) {
        $contents = file_get_contents($phpFile[0]);
        if(preg_match('/var_dump(\S*)/s', $contents) && !ctype_upper(end(explode('\\',$phpFi
            $where = $phpFile[0];
            $what=$contents;
            $chck = False;
        }
    }
    $this->assertTrue($chck,"".$where);
    }

    public function test_if_basic_tags_closed(){
        $dir = 'C:/wamp64/www/PISiBP-TravelApp/';
        $appDir = new RecursiveDirectoryIterator($dir);
        $iterator = new RecursiveIteratorIterator($appDir);
        $matches = new RegexIterator($iterator, '/^.+\.php$/i', RecursiveRegexIterator::GET_
        $chck = True;
        $where='';
        $what='';
        foreach ($matches as $phpFile) {

            $contents = file_get_contents($phpFile[0]);
            if(preg_match('/<html/', $contents)){
            if((!preg_match('/<\/body(.*?)>/', $contents) && !ctype_upper(end(explode('\\',$
                $where = $phpFile[0];
                $what=$contents;
                $chck = False;
                #$this->assertFalse(true);
            }
        }
        }
        $this->assertTrue($chck,"".$where);
        }
}
?>
```

Prva metoda proverava da li u okviru HTML koda postoji neželjeni PHP output. Druga metoda proverava da li su body tagovi zatvoreni. Putanja do fajla za testiranje se nalazi uz pomoć RegexIterator-a, nakon čega se čita sadržaj fajla i proverava da li ispunjava uslove.

2.4 InputTest.php

Fajl Input.php sadrži klasu Input koja služi za upravljanje HTTP zahtevima. Sastoji se od statičkih metoda exists i get koje proveravaju da li HTTP zahtev postoji, i vraćaju vrednosti iz asocijativnih nizova POST i GET, respektivno.

```
<?php
<?php
class Input
{
    public static function exists($type = 'post')
    {
        $type = strtolower($type);
        switch ($type) {
            case 'post':
                return (!empty($_POST)) ? true : false;
                break;
            case 'get':
                return (!empty($_GET)) ? true : false;
                break;
            case 'put':
                return (!empty($_PUT)) ? true : false;
                break;
            case 'head':
                return (!empty($_HEAD)) ? true : false;
                break;
            case 'cookie':
                return (!empty($_COOKIE)) ? true : false;
                break;
            case 'files':
                return (!empty($_FILES)) ? true : false;
                break;
            case 'env':
                return (!empty($_ENV)) ? true : false;
                break;
            case 'request':
                return (!empty($_REQUEST)) ? true : false;
                break;
            case 'session':
                return (!empty($_SESSION)) ? true : false;
                break;
            default:
                throw new Exception('Ne postoji takav zahtev');
                break;
        }
    }
    public static function get($item)
```

```
    {
        if (isset($_POST[$item])) {
            return $_POST[$item];
        } elseif (isset($_GET[$item])) {
            return $_GET[$item];
        }
        return '';
    }
}
?>
```

Potrebno je testirati da li statička metoda exists radi sa svim tipovima HTTP zahteva.

```
<?php
public static function provideRequestMethods()
    {
        return [
            ['POST'],
            ['GET'],
            ['PUT'],
            ['HEAD'],
            ['COOKIE'],
            ['FILES'],
            ['ENV'],
            ['REQUEST'],
            ['SESSION']
        ];
    }
        /**
    * @dataProvider provideRequestMethods
    */
    public function test_request_methods($req)
    {
        $this->assertContains(Input::exists($req), [true,false], $req . "");
    }
?>
```

Prosleđujući listu niskih koje predstavljaju nazive različitih HTTP zahteva statičkoj funkciji exists proverava se da li se ona ponaša očekivano kada su ulazni podaci u skupu očekivanih, ali UPPERCASE, ukoliko se funkcija ponaša očekivano, vratiće Boolean vrednost iz skupa [0,1], u drugom slučaju, doćiće do izuzetka sa porukom da ne postoji takav zahtev.

```php
<?php
public function test_fake_request_methods()
    {
        $this->expectExceptionMessage('Ne postoji takav zahtev');
        Input::exists('stuff');
    }
?>
```

Potrebno je proveriti da li za nepostojeći ulaz dolazi do prethodno navedenog izuzetka.

```php
<?php
public function test_input_post_values()
    {
        $_POST['ime'] = 'Djordje';
        $this->assertSame('Djordje', Input::get('ime'));
    }
    public function test_input_get_values()
    {
        $_POST = array();
        $_GET['ime'] = 'Djordje';
        $this->assertSame('Djordje', Input::get('ime'));
    }
?>
```

Na prazne nizove POST i GET se dodaje string sa nasumičnim karakterima kao vrednost za neki ključ, u ovom slučaju par ključ-vrednost ime-nasumično ime, i za ovakav ulaz proveriti da li metoda get radi očekivano.

```php
<?php
public function test_input_get(){
        $_POST = array();
        $_GET = array();
        $this->assertSame('', Input::get(random_bytes(256)));
    }
?>
```

Test identičan prethodnom, samo za ključ nasumično generisan dužine 256 karaktera, koji nema vrednost na koju pokazuje. Takođe, oba asocijativna niza su prazna na kraju testa kako ne bi došlo do nepotrebnih anomalija. Očekivani rezultat je prazna niska, koja govori o nepostojanju vrednosti za uneti ključ.

2.5 PonudeTest.php

Fajl PonudeTest.php sadrži klasu PonudeTest čije metode proveravaju integritet i vrednosti redova u tabeli **aranzmani**. U toku generisanja podataka, normalnom distribucijom umnoženom faktorom broja zvezdica smeštaja dobija se kapacitet hotela. Potrebno je proveriti, da li data funkcija obezbeđuje da ne postoje anomalije poput negativnih vrednosti za kapacitet nekog hotela, i da li je ukupni broj unetih aranžmana zadovoljavajuć, kao i broj prošlih i tekućih.

```php
<?php

 public function getSmestaj(){

        $db = DB::getInstance();
        $sql = "SELECT *
        FROM smestaj;";
        $res = $db->query($sql)->results();
        return $res;
    }
    public function getPonuda()
    {
        $db = DB::getInstance();
        $sql = "SELECT *
        FROM aranzmani
        GROUP BY smestaj_id
        HAVING MAX(aran_id);";
        $res = $db->query($sql)->results();
        return $res;
    }
    public function getAllPonude()
    {
        $db = DB::getInstance();
        $sql = "SELECT *
        FROM aranzmani;";
        $res = $db->query($sql)->results();
        return $res;
    }
    public function getAllPonudeWhere($cnd)
    {
        $db = DB::getInstance();
        $sql = "SELECT *
        FROM aranzmani WHERE " . $cnd;
        $res = $db->query($sql)->results();
        return $res;
    }
    public function validateDate($date, $format = 'Y-m-d H:i:s')
    {
```

```
        $d = DateTime::createFromFormat($format, $date);
        return $d && $d->format($format) == $date;
    }

?>
```

Funkcije iznad predstavljaju uslužne funkcije kojima se dolazi do informacija potrebnih za testiranje.

```
<?php
public function test_if_titles_correct(){
        $tester = true;
        $ponude = $this->getPonuda();
        $months = ["Januar","Jun","Jul","Avgust","Septembar","Oktobar"];
        $godine = [2022,2023];
        $prevoz = ["avionom","autobusom","brodom","samostalni prevoz","vozom"];
#Šangaj Januar 2023 ShangriLa Palace avionom 3 dana
        $dani = [3,5,7,10,14];
        foreach ($ponude as $ponuda) {
            $toCheck = explode(" ", $ponuda->naziv);
            if (in_array($toCheck, $months) || in_array($toCheck, $godine) || in_array($toCh
                $tester = false;
                break;
            }
        }
        $this->assertTrue($tester, "Greska u elementima iz ponude");
    }
?>
```

Svaki aranžman, tj. njegov naslov ili ime bi trebalo da prati konvenciju definisanu komentarom u testu, *Grad-Mesec-Godina-Naziv smeštaja-Prevozno sredstvo-Koliko aranžman traje*. Proverom da li svako polje svakog reda tabele ispunjava definisani kriterijum garantuje se integritet podataka aranžmana.

```
<?php
    public function test_if_dates_correct(){
        $tester = true;
        $ponude = $this->getPonuda();
        $id = -1;
        foreach ($ponude as $ponuda) {
            if (!($this->validateDate($ponuda->krece)) || !($this->validateDate($ponuda->vra
                $tester = false;
                break;
            }
        }
        $this->assertEquals(true, $tester, "Greska u datumima iz ponude, (aran_id=" . $id .
    }
?>
```

U toku generisanja podataka je moguće doći do unošenja pogrešnog formata datuma. Iako je mogućnost takvog ishoda minimalna, potrebno je proveriti.

```
<?php
public function test_if_populated()
    {
        $res = $this->getAllPonude();
        $this->assertGreaterThanOrEqual(60000, count((array)$res), count((array)$res) . "");
    }
    public function test_if_invalid_populated()
    {
        $res = $this->getAllPonudeWhere("krece<now()");
        $this->assertGreaterThanOrEqual(10000, count((array)$res), count((array)$res) . "");
    }
    public function test_if_valid_populated()
    {
        $res = $this->getAllPonudeWhere("krece>now()");
        $this->assertGreaterThanOrEqual(50000, count((array)$res), count((array)$res) . "");
    }
?>
```

Prethodne tri funkcije proveravaju da li je baza generisana shodno kriterijumima zadatka. Da li postoji više od 60000 aranžmana, od kojih su više od 50000 aktivnih i barem 10000 prošlih. Funkcija now() se koristi kako bi se argument pretraživanja odnosio na trenutni datum i vreme.

```
<?php
    public function test_if_capacity_optimal(){
        $check=True;
        $res = $this->getSmestaj();
        foreach($res as $row){
            if($row->kapacitet<=10){
                $check = False;
                break;
            }
        }
        $this->assertTrue($check);
    }
?>
```

U opisu ove tačke definisan je problem normalne distribucije kao i koje posledice može imati. Za generisanje zvezdica hotela korišćena je normalna distribucija sa osiguranom donjom granicom 3. Ovo nije slučaj za generisanje kapaciteta, tako da je potrebno proveriti ishod generisanja te tačke.

2.6 RedirectTest.php

Fajl Redirect.php sadrži statičku funkciju koja služi za usmeravanje korisnika na zadati link. Kao takva, bez ikakvih povratnih vrednosti i jednim izuzetkom, ne može da se reprodukuje u test okruženju(header-i...). Ideja je da se izvrši Mock-ovanje ili Stub-ovanje ove funkcije. PHPUnit ne podržava Mock-ovanje statičkih funkcija tako da dolazi do problema. U okviru iste klase dodata je funkcija *mimicTo* koja oponaša funkciju *to* tako što zadržava sve uslove i izuzetke, samo umesto usmeravanja korisnika vraća link na koji bi se korisnik usmerio.

```php
<?php
class Redirect
{
    public static function to($location = null){
        if ($location) {
            if (is_numeric($location)) {
                switch ($location) {
                    case 404:
                        header('HTTP/1.0 404 Not Found');
                        include 'includes/errors/404.php';
                        break;
                    default:
                        throw new RuntimeException();
                        break;
                }
            }
            header('Location: ' . $location);
            exit();
        }
    }
    #Uprostena prethodna funkcija->phpunit ne supportuje static fn mock/stub
    public static function mimicTo($location = null){
        if ($location) {
            if (is_numeric($location)) {
                switch ($location) {
                    case 404:
                        return 404;
                    default:
                        throw new RuntimeException();
                }
            }
            return $location;
        }
    }
}
?>
```

```php
<?php

declare(strict_types=1);

require_once("core/init.php");

use PHPUnit\Framework\TestCase;

class RedirectTest extends TestCase
{
    public static $headers;

    public function test_redirect_no_param()
    {
        $this->assertNotTrue(Redirect::to());
    }
    public function test_redirect_num_param()
    {
        $this->expectException(RuntimeException::class);
        Redirect::to(2023);
    }

    public static function providePossibleScenarios()
    {
        return [
            [404],
            ['neki_location'],
        ];
    }
    /**
    * @dataProvider providePossibleScenarios
    */
    public function test_mimic_redirect_multiple_param($p)
    {
        $this->assertSame($p, Redirect::mimicTo($p));
    }
    public function test_mimic_redirect_exception()
    {
        $this->expectException(RuntimeException::class);
        Redirect::mimicTo(403);
    }
}
?>
```

U ovakvom slučaju je moguće jednostavno proveriti ponašanje funkcije.

2.7 SessionTest.php

Fajl Session.php sadrži klasu Session koja sadrži statičke metode čitanja i manipulacije SESSION asocijativnog niza.

exists Proverava postojanje ključa prosleđenog kao parametar u nizu SESSION.

put Za argumente ključ-vrednost unosi takav par u niz SESSION.

get Za argument ključ vraća vrednost iz asocijativnog niza SESSION.

delete Za prosleđeni ključ briše par iz SESSION niza.

flash Omogućava istovremeno čitanje, vraćanje i brisanje elemenata SESSION čiji ključ odgovara argumentu funkcije ukoliko argument postoji u nizu, u suprotnom unosi taj ključ sa prosleđenom vrednošću.

```
<?php
declare(strict_types=1);
require_once("core/init.php");
use PHPUnit\Framework\TestCase;
class SessionTest extends TestCase{

public function test_session_put(){
        Session::put("ime", "Djordje");

        $this->assertTrue(Session::exists('ime'));
    }
    public function test_session_delete(){
        Session::delete("ime");
        $this->assertFalse(Session::exists('ime'));
    }
    public function test_flash_append(){
        Session::flash('ime', 'Djordje');
        $this->assertTrue(Session::exists('ime'));
    }
    public function test_flash_pop(){
        Session::flash('ime', 'Djordje');
        $this->assertFalse(Session::exists('ime'));
    }
    }
}
?>
```

Funkcije iznad proveravaju ponašanja metoda klase Session

2.8 TokenTest.php

Fajl Token.php sadrži klasu Token koja služi za generisanje i validaciju tokena.

Sadrži dve metode, *generate*, koja generiše, i *check* koja za token dat u argumentu proverava validnost.

```
<?php
class Token{

    public static function generate(){
        return Session::put(Config::get('session/token_name'),
         password_hash(random_bytes(256), PASSWORD_BCRYPT));
    }
    public static function check($token){
        $tokenName = Config::get('session/token_name');
        if (Session::exists($tokenName) && $token === Session::get($tokenName)) {
            Session::delete($tokenName);
            return true;
        }
        return false;
    }
}

?>
```

```
<?php
declare(strict_types=1);
require_once("core/init.php");
use PHPUnit\Framework\TestCase;
class TokenTest extends TestCase{

    public function test_token_integrity(){
        $new_token = Token::generate();
        $isIt = Token::check($new_token);
        $this->assertTrue($isIt);
    }
    public function test_fake_token(){
        $fake_token = password_hash(random_bytes(256), PASSWORD_BCRYPT);
        $isIt = Token::check($fake_token);
        $this->assertFalse($isIt);
    }
}
?>
```

Neophodno je testirati ponašanje metoda u slučajevima kada proveravaju validan token, i kada proveravaju nepostojeći token.

2.9 UserTest.php

Fajl User.php sadrži klasu User čije metode vrše CRUD operacije nad redovima u tabeli korisnik, zaposleni i admin. Cilj testiranja ove klase jeste provera izvršavanja svake od metoda, verifikacija rezultata i ponašanje usled neočekivanih ulaznih pdoataka.

```
<?php
public function duplicateTableUser(){

        $this->deleteTableUser();
        $sql = "CREATE TABLE korisnikFake(korisnik_id INT AUTO_INCREMENT PRIMARY KEY,
         tip VARCHAR(255))";
         $db = DB::getInstance();
         $db->query($sql);
    }
    public function deleteTableUser(){
        $sql = "DROP TABLE IF EXISTS korisnikFake";
        $db = DB::getInstance();
        $db->query($sql);
    }
?>
```

Uslužne metode za testiranje ove klase. Lažna tabela se koristi u svrhu testiranja jedne od metoda klase.

```
<?php
public static function provideFalseInfo(){
        return [
            ['admin',[]],
            ['worker',[1,2]],
            ['randomstring',[3,4]],
            [0,'0'],
        ];
    }
    /**
    * @dataProvider provideFalseInfo
    */
    public function test_if_create_foolproof($tip, $arr){
        $this->expectException(Exception::class);
        $user = new User();
        $user->create($tip, $arr);
    }
?>
```

Za niz lažnih podataka proveriti da li metoda create propušta neočekivani ulaz.

```php
<?php
  public function test_if_finds_fake_table()
    {
        $this->deleteTableUser();
        $this->duplicateTableUser();
        $user = new User();
        $this->expectExceptionMessage('Nepostojeca tabela');
        $table = "korisnikFake";
        $fields = ["tip" => "string", "rand" => "neki"];
        $user->create($table, $fields);
        $this->deleteTableUser();
    }
?>
```

Proverava da li metoda za unos korisnika dozvoljava korišćenje spoljnih tabela.

```php
<?php
public static function provideInvalidCredentials(){
        return [
            ['invalid@mail.address','123'],
            ['',''],
            ['randomstring','123456789'],
            [0,'0'],
            ['129000dd@ds..','4096543####1;']
        ];
    }
    /**
    * @dataProvider provideInvalidCredentials
    */
    public function test_login_invalid_credentials($username, $pw){
        $user = new User();
        $response = $user->login($username, $pw);
        $this->assertEquals($response, false);
    }
?>
```

Za nepostojeće parove adresa i lozinki, očekivani izlaz metode login je Boolean False.

```php
<?php
public function test_permission_level(){
        $user = new User();
        $this->assertNull($user->permissionLevel());
    }
?>
```

Ovaj test se bavi proverom permisija korisnika koji je tek napravljen-još nije unet u bazu podataka pa je očekivani izlaz null.

```php
<?php
public function test_login_valid_credentials(){
        $user = new User();
        $response = $user->login('djordje00karisic@gmail.com', '12345678');
        $this->assertEquals($response, true);
    }
?>
```

Za validne argumente se proverava rezultat metode login, očekivano je Boolean True.

```php
<?php
 public function test_logged_not_in_db_update(){
        $user = new User();
        $DBMock = $this->getMockBuilder('DB')->getMock();
        $DBMock->expects($this->any())->method('updateUser')->will($this->returnValue(false)
        $user->login('djordje00karisic@gmail.com', '12345678');
        $user->setDB($DBMock);
        $this->expectExceptionMessage('Desio se problem tokom azuriranja.');
        $user->update('admin', 1, [1]);
    }
?>
```

Ovaj test se služi Mock-ovanjem kako bi se testiralo svojstvo metode updateUser koje po default-u vraća rezultat SQL komande koja se izvršava van dometa test okruženja, tako što se ta metoda 'modifikuje' da uvek vraća false, kako bi se simuliralo neuspešno upisivanje u bazu.

```php
<?php
 public static function provideUserInfoForFind(){
        $db = DB::getInstance();
        $res = $db->query("SELECT korisnik_id,tip FROM `korisnik` WHERE korisnik_id in (SELE
        $idN = $res->first()->korisnik_id;
        $res = $db->query("SELECT email FROM `admin` WHERE korisnik_id =?;", $params = [$idN
        $email = $res->first()->email;
        return [[$idN],[$email]];
    }
    /**
    * @dataProvider provideUserInfoForFind
    */
    public function test_user_exists($idN){
        $user = new User();
        $exists = $user->find($idN);
        $this->assertEquals(true, $exists);
    }
?>
```

Za postrojeći red podataka iz baze korisnik, proveravaju oba use case-a metode *find*. Za ovakve podatke očekivani izlaz je Boolean True.

```php
<?php
public function test_if_user_information(){
        $this->deleteTableUser();
        $user = new User();
        $check = $user->login('djordje00karisic@gmail.com', '12345678');
        $this->assertNotEquals(false, $check);
    }
?>
```

Provera metode login za važeću listu podataka. Očekivani izlaz je sve osim Boolean False, koji se vraća ukoliko metoda nije izvršena kako treba.

```php
<?php
 public function test_database_error_insert()
    {
        $DBMock = $this->getMockBuilder('DB')->getMock();
        $DBMock->expects($this->any())->method('insert')->will($this->returnValue(false));
        $user = new User();
        $this->expectExceptionMessage('Desio se problem tokom kreiranja naloga.');
        $user->create('zaposleni', [1,2], $DBMock);
    }
?>
```

Ovaj test koristi tehniku Mock-ovanja kako bi simulirao neuspešno izvršen 'query' unošenja korisnika u bazu podataka i proverio da li zaista poziva izuzetak.

```php
<?php
public function test_logout(){
        $user = new User();
        $user->login('djordje00karisic@gmail.com', '12345678');
        $user->logout();
        $this->assertFalse($user->isLoggedIn());
    }
?>
```

Testiranje metode 'logout', ujedno i brisanje sesije koja je napravljena za vreme testa.

2.10 ValidateTest.php

Fajl Validate.php sadrži klasu Validate čiji se objekat bavi konstruisanjem setova pravila unosa podataka u bazu, kako bi se proverio korisnički unos, i eventualno prikazao grešku ukoliko je unos nepotpun.

```
<?php
public function test_returns_dbinst()
    {
        $db = DB::getInstance();
        $val = new Validate();
        $this->assertSame($db, $val->getDB());
    }
?>
```

Provera da li ova klasa poštuje princip obrasca Singleton.

```
public function test_validate_rules_valid_data(){
        $validate = new Validate();
        $validation = $validate->check(
            array('password_current' => 'djordje'),
            array('password_current' => array('required' => true,'min' => 6
            ),)
        );
        $this->assertEmpty($validate->errors());
    }
<?php
?>
```

Za simulirani pravilni unos korisnika, i podešeni set pravila koji on mora poštovati, očekivani rezultat je da nema greške.

```
<?php
public function test_validate_rules_valid_data()
    {
        $validate = new Validate();
        $validation = $validate->check(
            array('password_current' => 'djordje'),
            array('password_current' => array('required' => true,'min' => 6
            ),)
        );
        $this->assertEmpty($validate->errors());
    }
    public static function provideInvalidData(){
        return [[array('password_current' => 'djordje','password_new' => ''),array(
        'password_current' => array('required' => true,'min' => 6),'password_new' =>
        array('required' => true,'min' => 6))],[
        array('password_current' => 'djordje','password_new' => 'ab'),
```

```
        array('password_current' => array('required' => true,'max' => 5),
        'password_new' => array('required' => true,'max' => 3))],
        [ array('email' => 'djordje00karisic@gmail.com','password' => '12345678',
        'new_password' => '12345678'),array('email' => array('required' => true,
        'numeric' => true,'unique' => 'admin'),'password' => array(
        'required' => true,'numeric' => true,))],
        [ array('password_current' => '12345678','password_new' => '123456789',
        'password_new_again' => '12345678'),
        array('password_current' => array('required' => true,
        'min' => 6),'password_new' => array('required' => true,'min' => 8),
        'password_new_again' => array('required' => true,'min' => 8,
        'matches' => 'password_new'))]];
    }
/**
    * @dataProvider provideInvalidData
    */
    public function test_validate_rules_invalid_data($data, $test){
        $validate = new Validate();
        $validation = $validate->check($data, $test);
        $this->assertNotEmpty($validate->errors());
    }
?>
```

Za definisani set lažni, nevažeći podataka i pravila proveriti da li objekat klase Validate popunjava niz grešaka.

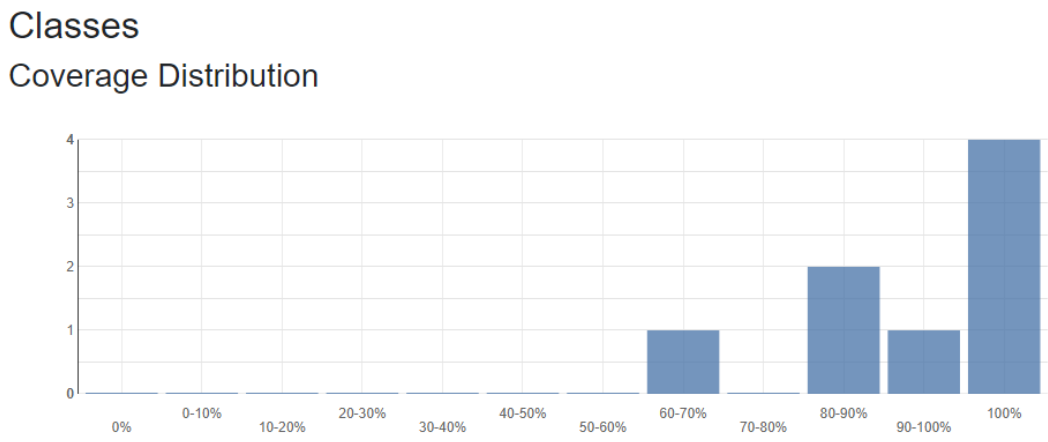
```
<?php
public function test_validate_rules_valid_credentials(){
        $validate = new Validate();
        $validation = $validate->check(
            array('email' => 'djordje00karisic@gmail.com','password' => '12345678'),
            array(
            'email' => array(
                'required' => true,
                'unique' => 'admin'),
            'password' => array(
                'required' => true,
                'numeric' => true,)));
            $this->assertEmpty($validate->passed());
    }
?>
```

Za validan skup podataka i pravila kao ulaz, proveriti da li je validacija uspešna preko funkcije 'passed'. Očekivani izlaz je prazan skup ili Boolean false zato što je kao pravilo uneto da u bazi ne sme postojati polje sa vrednošću zadatim kao email, što nije ispunjeno zbog postojanja takvog korisnika.

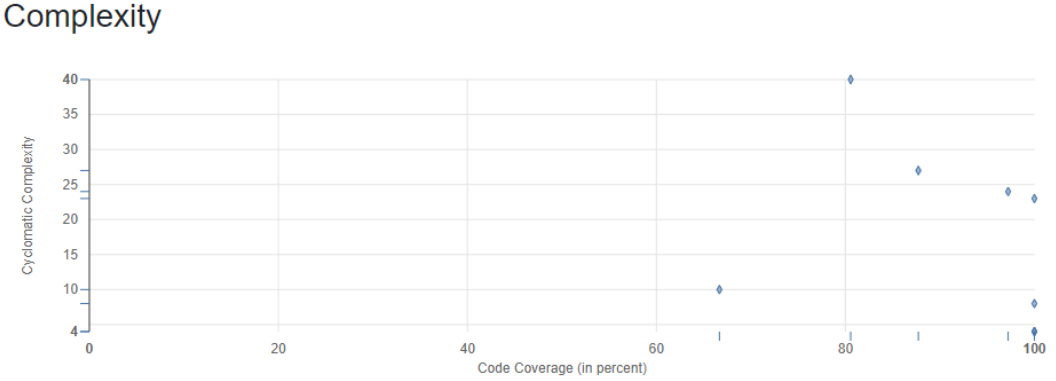
2.11 Generisanje izveštaja QA analize

U uvodu je pomenuto da će se kao alat za generisanje izveštaja koristiti PHPUnit metoda coverage sa parametrom html, koja generiše celokupnu analizu sa pratećim graficima kao HTML5 dokument. U ovoj sekciji biće prikazan celokupan izgled ove stranice.

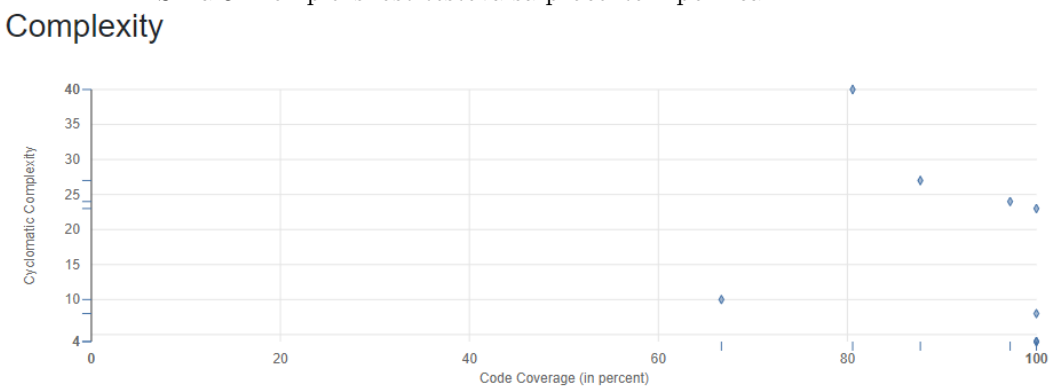
Slika 1: Procenat pokrivanja klasa



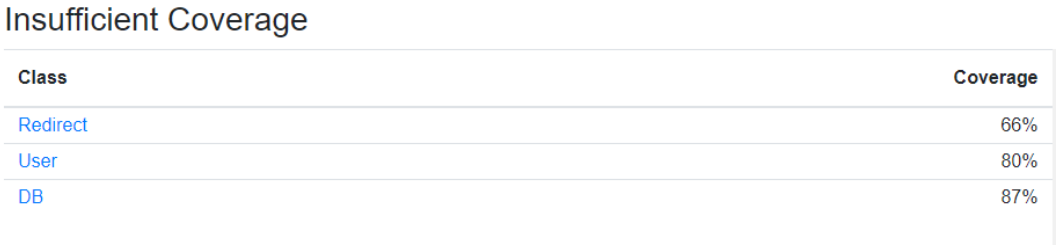
Slika 2: Kompleksnost testova sa procentom pokrića



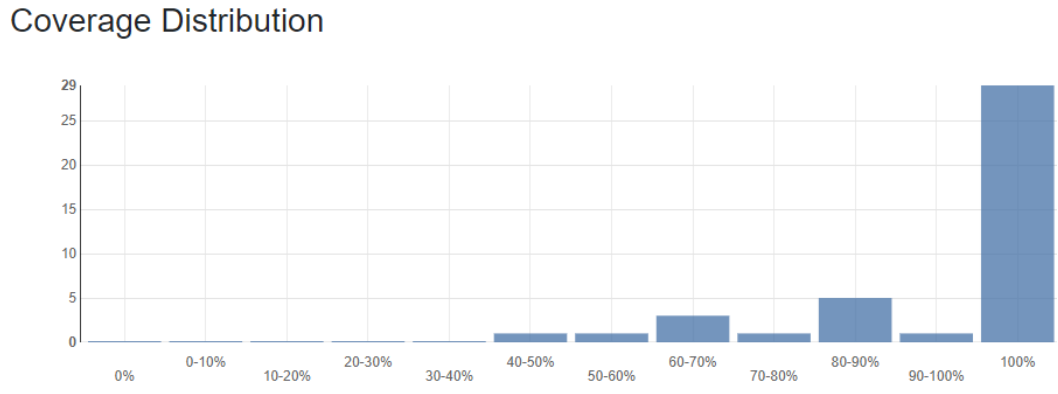
Slika 3: Kompleksnost testova sa procentom pokrića



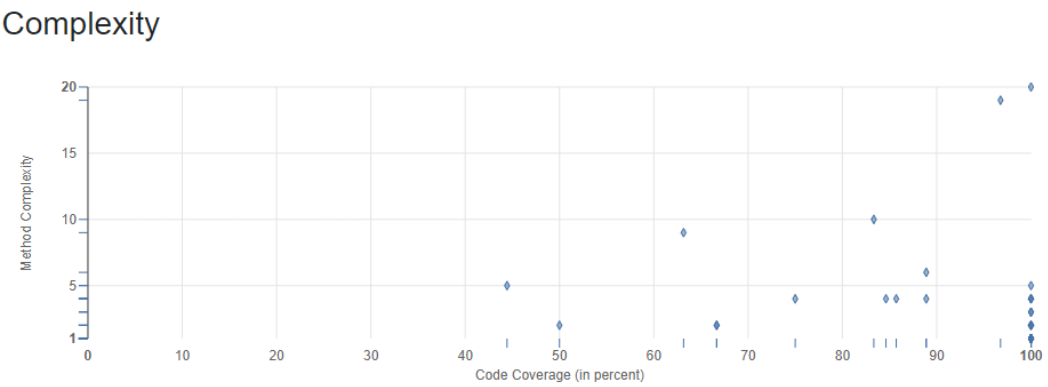
Slika 4: Klase koje nisu u potpunosti testirane, razlog dat u sekcijama tih klasa



Slika 5: Procenat pokrića metoda klasa, niži procenat imaju metode koje koriste metode objekata van opsega testne klase, rešeno mock-ovanjem, međutim softver za generisanje pokrivenosti ne vidi rešenje



Slika 6: Procenat pokrića metoda klasa, sa respektivnom kompleksnošću testova koji ih pokrivaju



Slika 7: Konačni prikaz pokrivenosti projekta testovima, razlog iza crvenih polja objašnjen u okviru sekcija za klasu.

	Code Coverage							
	Lines			Functions and Methods			Classes and Traits	
Total		88.26%	203 / 230		70.73%	29 / 41		50.00%	4 / 8
Config.php		100.00%	7 / 7		100.00%	1 / 1		100.00%	1 / 1
DB.php		87.72%	50 / 57		58.33%	7 / 12		0.00%	0 / 1
Input.php		100.00%	25 / 25		100.00%	2 / 2		100.00%	1 / 1
Redirect.php		66.67%	10 / 15		50.00%	1 / 2		0.00%	0 / 1
Session.php		100.00%	10 / 10		100.00%	5 / 5		100.00%	1 / 1
Token.php		100.00%	7 / 7		100.00%	2 / 2		100.00%	1 / 1
User.php		80.82%	59 / 73		54.55%	6 / 11		0.00%	0 / 1
Validate.php		97.22%	35 / 36		83.33%	5 / 6		0.00%	0 / 1

3 Python-Generisanje podataka

Biblioteke korišćene za generisanje, struktuiranje i čuvanje podataka su sledeće:

Selenium Prikupljanje podataka sa web stranica.

Pandas Skladištenje i manipulacija podataka.

Numpy Operacije za nizove podataka (series,dataframe).

Faker Generisanje jednostavnih lažnih podataka.

OpenAI Generisanje složenijih lažnih podataka uz pomoć davinci-003 modela.

matplotlib Predstavljanje podataka graficima.

math matematičke funkcije.

random Generisanje nasumičnih brojeva.

re Biblioteka za korišćenje Regex šablona.

bs4-beautiful soup4 Reprezentacija HTML objekta.

time Biblioteka za predstavljanje vremena kao objekat.

translators, translators.server,srtools Prevod podataka.

bing-image-downloader Nalaženje i čuvanje slika za podatke.

utils Lokalna biblioteka koja sadrži metode koje vrše operacije nad podacima.

VAŽNA NAPOMENA: Izvršavanje nekih metoda sledećih biblioteka(OpenAI,translators...) može potrajati ili čak biti prekinute, jer zavise od gustine saobraćaja korisnika koji su povezan na API, tako da generator.py neće biti korišćen u okviru Docker kontejnera, osim ukoliko je isključivo neophodno. Detaljni opisi logike svih funkcija neće biti dati ovim izveštajem.

3.1 generator.py

Fajl generator.py predstavlja skriptu koja sadrži metode isključivo za generisanje i podataka uz pomoć pratećih biblioteka - Data Mining, Web Scraping, Data Analysis, Data Wrangling...

```
link = 'https://www.worldometers.info/geography/7-continents/'

PATH_DATA = "pydata/py_csv/"
```

link promenljiva sadrži putanju do web stranice sa koje je potrebno uzeti podatke o potrebnoj 'geografiji'. PATH-DATA predstavlja putanju do foldera na kojem se čuvaju preuzeti podaci, u formatu csv.

```
def getDriver():

    browser = webdriver.Chrome()
    return browser
```

Metoda za kreiranje instance Chrome driver-a.

```
def getAllGeography():
    kontinenti,drzave = getContinentsAndCountries()
    newlink = 'https://worldpopulationreview.com/countries/cities/'
    browser = getDriver()
    gradovi = pd.DataFrame(columns = ['naziv','drzava','kontinent'])
    for name in drzave['naziv'].values:
        clink = newlink + name.lower().replace(' ','-')
        browser.get(clink)
        time.sleep(3)
        content = browser.page_source
        soup = BeautifulSoup(content,features="lxml")
        try:
            browser.find_element(By.CLASS_NAME('_3p_1XEZR')).submit()
        except:
            pass
        for a in soup.findAll('table', attrs={'class':'table'}):
            howmuch = 5 if drzave.loc[drzave["naziv"]==name,"kontinent"].values[0] == 'Europ
            for city in a.findAll('td'):
                if not howmuch:
                    break
                if all(x.isalpha() or x.isspace() for x in city.text):
                    gradovi.loc[len(gradovi.index)] = [city.text,drzave.loc[drzave['naziv']=
                    howmuch-=1
    gradovi.to_csv(PATH_DATA+'cities.csv',index=None)
    gradovi['naziv']=gradovi['naziv'].apply(translateElement)
    gradovi['drzava']=gradovi['drzava'].apply(translateElement)
    gradovi['kontinent']=gradovi['kontinent'].apply(translateElement)
```

```
    gradovi.to_csv(PATH_DATA+'gradovi.csv',index=None)
    return gradovi,drzave,kontinenti
```

Primer metode koja koristi Selenium u kombinaciji sa bs4 kako bi našla potrebne podatke, i prosledila ih kao povratnu vrednost. Postoji više ovakvih funkcija, međutim radi prostora na dokumentu je dovoljno prikazati samo ovu.

```
def getAllDFs():
    gradovi,drzave,kontinenti = getAllGeography()
    hoteli = pd.DataFrame(columns=["naziv","grad"])
    for index, val in gradovi['naziv'].items():

        imena = hotelGPT(val)
        for ime in imena:
            hoteli.loc[len(hoteli.index)] = [ime,val]


    hoteli.to_csv(PATH_DATA+'hoteli.csv',index=None)

    return hoteli,gradovi,drzave,kontinenti
```

Ova funkcija unutar sebe sadrži poziv više funkcija koje nalaze potrebne podatke i vraćaju kao return value. Nakon ove funkcije u okviru foldera py-csv postoje svi potrebni podaci vezani za gradove, države i kontinente i hotela. Imena hotela se generišu uz pomoć uslužne funkcije hotelGPT koja će biti opisana u okviru utils.py datoteke.

```
def generateRooms():   #testirati sumu broja soba
    hoteli = pd.read_csv(PATH_DATA+'hoteli.csv')
    np.random.seed(5)
    mu = 75
    sigma = 25
    hoteli['br_soba'] = np.random.normal(mu, sigma, hoteli.shape[0])*hoteli['zvezdice']
    hoteli['br_soba']=hoteli['br_soba'].apply(round)
    hoteli.to_csv(PATH_DATA+'hoteli.csv',index = None)
```

Funkcija generisanja kapaciteta smeštaja, koristi se normalna distribucija sa srednjom vrednošću μ i standardnim odstupanjem σ. Rezultat ove funkcije se pokazao kao optimalan osim u slučaju anomalije kada distribucija vrati negativnu vrednost, što je kontrolisano testovima.

```
def hotelStarsDistribution():
    np.random.seed(5)
    mu = 3.5
    sigma = 0.5
    try:
        hoteli = pd.read_csv(PATH_DATA+'hoteli.csv')
```

```
    except:
        hoteli,_,_,_ = getAllDFs()
    hoteli['zvezdice'] = np.random.normal(mu, sigma, hoteli.shape[0])
    hoteli['zvezdice'] = hoteli['zvezdice'].apply(roundStars)
    hoteli.to_csv(PATH_DATA+'hoteli.csv',index = None)
```

Funkcija identična prethodnoj, samo za broj zvezdica hotela. Donja granica broja zvezdica je 3. Predstavljeno osnovom 0.5, radi raznovrsnijeg izbora smeštaja.

```
def hotelGenerateAddress():
    try:
        hoteli = pd.read_csv(PATH_DATA+'hoteli.csv')
    except:
        hoteli,_,_,_ = getAllDFs()
    fake = Faker()
    cutSpaces = np.vectorize(lambda x: x.replace('  ',' '))
    for y in x[0:len(x.split(' '))-1]]
    cutAptNums = np.vectorize(lambda x:
    (' '.join([''.join(y) for y in x.split(' ')[0:len(x.split(' '))-1]])
    if x.split(' ')[len(x.split(' '))-1].isnumeric() else x))
    addresses = cutAptNums(
    cutSpaces(np.array([' '.join([y if not
    ('Suite' in y or 'Apt.' in y) else '' for
    y in fake.address().split('\n')[0].split(' ') ])
    for i in range(len(hoteli.index))],dtype=str)))
        #this is why we love python
    hoteli['adresa'] = addresses
    hoteli.to_csv(PATH_DATA+'hoteli.csv',index=None)
```

Funkcija koja se koristi metodom Faker klase, address, koja generiše lažnu adresu, dodaje u DataFrame i čuva stanje.

```
def PlotGeneratedInfos():
    hoteli = pd.read_csv(PATH_DATA+'hoteli.csv')
    gradovi = pd.read_csv(PATH_DATA+'gradovi.csv')
    plotTools(what=hoteli,case=1,title="Raspodela zvezdica hotela",x="Broj zvezdica",y="Koli
    plotTools(what=gradovi,case=2,title="Raspodela kolicine hotela po zemljama",x="Drzava",y
    plotTools(what=gradovi,case=3,title="Raspodela kolicine hotela po kontinentima",x="Konti
    #plotTools(what=hoteli['zvezdice'],case=1,title="Raspodela zvezdica hotela",x="Broj zvez
    plotTools(what=hoteli,case=4,title="Raspodela soba po hotelima",x="Hoteli",y="Broj soba"
```

Ova funkcija generiše grafike koji ukazuju na odnose kolona DataFrame-ova, prikaz raspodele podataka.

```
def sobaParamGen(base=10):
    sobe = pd.DataFrame(columns=['tip','br_kreveta','opis','gen_cena'])
    for i in list(TYPE.keys()): #da vrati samo tip,broj kreveta, pocetnu cenu
        for j in BED_NUM:
            kreveti_multiplier = 1 if j<2 else 0.67
            sobe.loc[len(sobe.index)] = [i,j,TYPE[i][0],
            round(TYPE[i][1]*j*kreveti_multiplier * base,2)]
    sobe.to_csv(PATH_DATA+'sobe.csv',index = None)
```

Funkcija koja generiše sobe sa pratećim parametrima, i generičnu cenu koja se konstruiše tako što se uzima tip sobe i multiplier koeficijent koji zavisi od broja kreveta.

```
def generatePrevoznik():
    prevoz = pd.DataFrame(columns=['tip','tip_komp'])
    prevoz['tip'] = ['airplane', 'bus', 'cruise', 'train']
    prevoz['tip_komp'] = companyGPT()
    prevoz['cena'] = [random.choice(range(1000,3000)),
    random.choice(range(100,250)),
    random.choice(range(250,950)),random.choice(range(100,150))]
    prevoz.loc[len(prevoz.index)] = ['lično vozilo', 'samostalni prevoz',0]
    prevoz.to_csv(PATH_DATA+'prevoz.csv',index=None)
```

Ova funkcija se bavi generisanjem prevoznika, koristi se ponovo openAI model davinci-003 kako bi generisao optimalna imena za agencije prevoza. Cena je nasumični broj iz liste brojeva definisane ograničenjima projekta.

```
def generatePonude():
    """
    aranzman(aran_id,naziv,krece,vraca,nap,smestaj_id,p_id)

    naziv->

    """
    gradovi = pd.read_csv(PATH_DATA+'gradovi.csv')
    hoteli = pd.read_csv(PATH_DATA+'hoteli.csv')
    prevoz = pd.read_csv(PATH_DATA+'prevoz.csv')
    aranzman = pd.DataFrame(columns = ["naziv","krece","vraca","smestaj","p_id"])
    hoteli['tmp'] = 1
    prevoz['tmp'] = 1
    prevoz['p_id'] = prevoz.index+1
    prevoz['prevod'] = ["avionom","autobusom","krstarenje/brodom",
    "vozom","samostalni prevoz"]
    prevoz['tmp']=1

    temp=pd.DataFrame()
    temp['naziv'] = hoteli['naziv']
```

```
temp['smestaj_id'] = temp.index+1
aranzman = pd.merge(hoteli, prevoz, on=['tmp'])

def startDateGenerator(margin,min_date,max_date):
#uzmi random dan za start iz dana iz dates array gde vazi da zadnji
dan meseca-start nije vece od margin(trajanje putovanja)
    maxed_date   = datetime.datetime.strptime(max_date, '%Y-%m-%d') - datetime.timedelta
    #minimized_date = datetime.strptime(min_date, '%Y-%m-%d')
    + pd.DateOffset(days=margin)

    dates = pd.date_range(min_date,maxed_date,freq='d').to_list()

    start_date = random.choice(dates)
    #end_date = datetime.strptime(start_date, '%Y-%m-%d')
    + pd.DateOffset(days=margin)
    return start_date + datetime.timedelta
    (hours=random.choice(range(1,20)))
    def endDateGenerator(margin,start_date):
    return start_date +
    datetime.timedelta(days=int(margin)) +
    datetime.timedelta(hours=random.choice(range(0,3)))

timeline = pd.DataFrame()
timeline['broj_dana'],timeline['tmp'] = ['3','5','7','10','14'],1
meseci = pd.DataFrame()
meseci['mesecMin'],meseci['mesecMax'],meseci['tmp'] =
['2023-1-1','2023-6-1','2023-7-1','2023-8-1','2023-9-1','2023-10-1'],
['2023-1-31','2023-6-30','2023-7-31','2023-8-31','2023-9-30','2023-10-31'],1
meseci['mesec'],meseci['mpr'] = ['1','6','7','8','9','10'],
["Januar","Jun","Jul","Avgust",
"Septembar","Oktobar"]
aranzman = pd.merge(aranzman, timeline, on=['tmp'])
aranzman = pd.merge(aranzman, meseci, on=['tmp'])
aranzman=aranzman.merge(temp,on=['naziv'],how='left')
aranzman = aranzman.drop('tmp', axis=1)
aranzman['datum_pocetka'] = aranzman.apply(lambda x:
startDateGenerator(x['broj_dana'],x['mesecMin'],
x['mesecMax']),axis=1)
aranzman['datum_zavrsetka'] = aranzman.apply(lambda x:
endDateGenerator(x['broj_dana'],
x['datum_pocetka']),axis=1)
aranzman = aranzman.drop(columns=['mesecMax','mesecMin'],
axis=1)
aranzman['godina'] = aranzman['datum_pocetka'].dt.strftime('%Y')
aranzman['m_str'] = aranzman['datum_pocetka'].dt.strftime('%')
#hoteli.loc[hoteli['naziv']==aranzman['naziv'],"grad"].values() +
```

```
    aranzman['ime'] = aranzman['grad'] + " " + aranzman['mpr'] + " "+
    aranzman['godina'] + " " + aranzman['naziv'] + " " + aranzman['prevod']+
    " " + aranzman['broj_dana'] + " dana"
    aranzman.to_csv(PATH_DATA+"aranzmani.csv",index=None)
```

Funkcija generatePonude() uzima u obzir sve zahteve navedene u dokumentu zahteva projekta, i na osnovu istih generiše ponude. Data-Wrangling kako bi se podaci doveli do željenog formata.

```
def generateAktivnosti():
    df = pd.DataFrame(columns = ["naziv"])

    for x in ["Setnja po gradu","Obilazak nacionalnog parka","Poseta muzeju","Nocenje","Faku
                "Obilazak obliznjih lokaliteta","Organizovani nocni provod"]:
        df.loc[len(df.index)]=x;

    df.to_csv(PATH_DATA+"aktivnosti.csv",index=None);
```

Generisanje DataFrame-a koji sadrži tipove aktivnosti. Čuvanje istog u py-csv datoteci.

```
def smestajImaAktivnost():#g_id        akt_id        smestaj_id


    gradovi = pd.read_csv(PATH_DATA+"gradovi.csv").index+1
    akt = pd.DataFrame(columns = ["g_id","akt_id","smestaj_id"])
    akt['smestaj_id'] = pd.read_csv(PATH_DATA+"hoteli.csv").index+1
    akt=akt.fillna('0')
    akt['g_id'] = akt['g_id'].apply(dodajRandomGrad)
    akt['akt_id'] = akt['akt_id'].apply(dodajRandomAktivnost)
    akt.to_csv(PATH_DATA+"aktivnosti_u_gradu.csv",index=None)
    #akt['g_id'] =

def generateImaAktivnost():
    decompose = lambda x: [1 for i in range(int(x))]

    aran = pd.read_csv(PATH_DATA+"aranzmani.csv")
    aran['broj_dana'] = aran['broj_dana'].apply(lambda x: x-1)

    akt = pd.read_csv(PATH_DATA+"aktivnosti.csv")
    ids = akt.index+1
    ima = pd.DataFrame(columns=['aran_id','akt_id'])
    aran=aran.drop(columns=['naziv','grad','br_soba','adresa','tip','tip_komp',
    'p_id','prevod','mesec','mpr','datum_pocetka','datum_zavrsetka',
    'godina','m_str','ime','zvezdice'])
    aran['aran_id'] = aran.index+1
    aran['akt_id']=aran['broj_dana'].apply(decompose)
```

```
    ima = aran.explode('akt_id')
    ima = ima.drop(columns=['broj_dana','smestaj_id'])
    ima['akt_id'] = np.random.choice(ids, ima.shape[0])
    ima.to_csv(PATH_DATA+"ima_aktivnost.csv",index=None)

def generateRandomRezervacije(n):
    #ime prezime        br_kartice        email        broj_odr        broj_dece
    cena        kom        kontakt        aran_id        korisnik_id
    faker = Faker()
    rez = pd.DataFrame(columns=['ime','prezime','br_kartice',
    'email','broj_odr','broj_dece',
    'cena','kom','kontakt','aran_id'])
    rez['ime'] = [faker.name().split(" ")[0] for _ in range(n)]
    rez['prezime'] = [faker.name().split(" ")[1] for _ in range(n)]
    rez['br_kartice'] = [''.join(random.choices(string.ascii_lowercase, k=8))
    for _ in range(n)]
    rez['email'] = [faker.name().split(" ")[1]+"@gmail.com"
    for _ in range(n)]
    rez['broj_odr'] = [random.randint(0,3) for _ in range(n)]
    rez['broj_dece'] = [random.randint(0,3) for _ in range(n)]
    rez['cena'] = [random.randint(100,500) for _ in range(n)]
    rez['kom'] = [random.randint(1,5) for _ in range(n)]
    rez['kontakt'] = [''.join(random.choices(string.ascii_lowercase, k=10))
    for _ in range(n)]
    rez['aran_id'] = [random.randint(1,50000) for _ in range(n)]

    rez.to_csv(PATH_DATA+"rand_rez.csv",index=None)
```

Funkcije iznad su za zadatak imale povezivanje aranžmana sa određenim aktivnostima, i povezivanje smeštaja sa tačno jednom aktivnošću u drugom gradu, kako bi se ostvarilo da svaki aranžman ima $(brDana - 1) * rand(aktivnost) + this.smestaj.aktivnost$ aktivnosti. Ovo predlaže postojanje raznolikih aktivnosti čak i ako u bazi postoji samo 8 tipova. Takođe funkcija generateRezervacije kreira DataFrame koji sadrži nasumično generisane rezervacije, zarad testiranja i manipulacije tabele rezervacije u bazi podataka.

```
def generator():
    getAllDFs()
    hotelStarsDistribution()
    generateRooms()
    hotelGenerateAddress()
    sobaParamGen()
    PlotGeneratedInfos()
    generatePrevoznik()
    generatePonude()
    generateAktivnosti()
```

```
    smestajImaAktivnost()
    generateImaAktivnost()
    generateRandomRezervacije(150)
```

Poziv generatora ima redosled izvršavanja prikazan prethodnom sekcijom koda.

3.2 utils.py

Funkcije definisane u okviru fajla utils.py predstavljaju uslužne funkcije koje se koriste u okviru populator.py i generator.py fajlova. Omogućava zadovoljavajuću preglednost koda ovih fajlova tako što sav posao izvan spektra delegiraju ovoj biblioteci, npr. svi pozivi openAI modula se vrše u okviru ove biblioteke.

```
fr,to = 'en_US','sr-Cyrl_RS'

BED_NUM = [1,2,3,4]
TYPE = {
    'SUITE':["Soba u hotelu (dve prostorije koje ne moraju biti
    odvojene vratima) odgovarajuće kvadrature za kapacitet od
    četiri osobe sa minimum dva pomoćna ležaja
    ili sofom na otvaranje.Sadrži stabilnu internet konekciju,
    frižider, TV i AC.",1.75],
    'FAMILY':["Porodična soba (jedna prostorija) odgovarajuće
    kvadrature za četiri osobe sa minimum jednim pravim pravim i
    jednim pomoćnim ležajem ili sofom na
    otvaranje, za decu.Sadrži stabilnu internet konekciju, frižider,
    TV i AC.",1.5],
    "SUPERIOR ROOM":["Soba u hotelu veće kvadrature i kvalitetnije
    opreme od standardne. Sadrži stabilnu internet konekciju, frižider,
    TV, klimu i sobni sef.",2],
    "SOBA (STANDARDNA)":
    ["Smeštajna jedinica u hotelu ili vili koja nema
    kuhinjske elemente, ni terasu. Sadrži stabilnu internet konekciju i TV.",1],
    "STUDIO":["‘‘Garsonjera’’ - smeštajna jedinica u hotelu,
    bez predsoblja, u kojoj se u istom prostoru nalazi deo sa kuhinjskim
    elementima sa osnovnim priborom za jelo.Sadrži stabilnu internet konekciju,
    TV i frižider.",1.25]
}

def dodajRandomGrad(y):
    x = pd.read_csv(PATH_DATA+"gradovi.csv")
    return random.choice(range(1,len(x.index)+1))

def dodajRandomAktivnost(y):
    x = pd.read_csv(PATH_DATA+"aktivnosti.csv")
    return random.choice(range(1,len(x.index)+1))
#TYPE SADRZI OPIS I
```

```
# MULTIPLIERE ZA CENE SOBA
def roundStars(x,base=0.5):
    if x>=5:
        return 5
    if x<=3:
        return 3
    return base * round(x/base)
```

TYPE rečnik sadrži informacije o mogućim sobama. fr,to globalne promenljive nose informaciju o jezicima sa kojeg i na koji se prevodi kasnije. Funkcija dodajRandomGrad vraća ID nasumičnog grada tako što uzima indeks i inkrementuje za 1. Naredna funkcija je identična.

```
def translateElement(el):
    """Prevodi crawlovane podatke sa engleskog na srpski, latinica
    Args:
        el (str): mesto/rec na engleskom
    Returns:
        str: mesto/rec na srpskom, spakovano kao latinica
    """
    global fr,to
    return cyrillic_to_latin(tss.lingvanex(el, fr, to)['text'])
```

Prevod imena gradova, država i kontinenata. Procenat tačnosti nije optimalan, ali je dovoljan kako bi podaci bili jasni.

```
def dataframeCleaner(df,path):
    """Usluzna funkcija koja cisti dataframe-ove tako sto izbacuje prazne elemente
    Args:
        df (DataFrame): DataFrame koji je potrebno ocistiti
        path (str): Putanja
    Returns:
        DataFrame: Ociscen DataFrame
    """
    df = df.loc[df['naziv']!='']

    df.to_csv('pydata/'+path,index = None)

    return df
```

Jednostavna funkcija koja proverava i čisti DataFrame koji u okviru kolone naziv ima praznu vrednost.

```
def setGPT():
    """Poziva GPT3

    """
    with open('pydata/api.txt') as f:
        openai.api_key = f.readlines()[0]

def hotelGPT(place):
    """Poziva GPT3 kako bi generisao listu imena hotela za dati grad
    Args:
        place (str): Neki grad

    Returns:
        list: Listu stringova generisanih naziva hotela
    """
    setGPT()
    rgx = '[^a-zA-ZčćžšđČĆŽŠĐ ]+'
    prompt = "Generiši 3 nasumična imena za hotele u mestu "+place+" koji ne sadrže ime mest
    response = openai.Completion.create(
    engine="text-davinci-003",
    prompt=prompt,temperature=0.9,
    max_tokens=150,
    top_p=1,
    frequency_penalty=0.0,
    presence_penalty=0.6,)
    h_list = response.choices[0]["text"].split("\n")
    #REGEX TO THE RESCUE
    h_list = [re.sub(rgx, '', x).replace(' ','',1) for x in h_list if x!='']
    h_list = [x for x in h_list if x!='' and len(x)!=1]
    time.sleep(2)   #greedy OpenAI dozvoljava samo 60 entry-ja po minutu tako da procesor mo
    return h_list
#setGPT("Šangaj")
def companyGPT():
    setGPT()
    prompt = "Generiši 4 nasumična jednostavna imena kompanija za prevoz turista do turistič
    with open('pydata/api.txt') as f:
        openai.api_key = f.readlines()[0]


    response = openai.Completion.create(
    engine="text-davinci-003",
    prompt=prompt,temperature=0.9,
    max_tokens=150,
    top_p=1,
    frequency_penalty=0.0,
    presence_penalty=0.8,)
```

```
    rgx = '[^a-zA-Z ]+'
    h_list = response.choices[0]["text"].split("\n")
    h_list = [re.sub(rgx, '', x).replace(' ','',1) for x in h_list if x!='']
    h_list = [x for x in h_list if x!='' and len(x)!=1]
    return h_list
```

Prva funkcija čita API ključ definisan u txt fajlu u okviru projekta.

Druga funkcija pomoću openAI-ja generiše imena hotela i uređuje da odgovara predefinisanim zahtevima DataFrame-u u kojem će biti dodati. Regex šablon omogućava čuvanje imena koja su generisana srpskom latinicom. Identična funkcija ispod ove za zadatak ima generisanje imena prevoznika.

```
def plotTools(case,what,
              title='',x='',y='',path=''):
    """Plotuje raspodelu NORMALNO rasporedjenih
    elemenata u pd.series-u

    Args:
        what (pd.series): Sta se plotuje
        sigma (float): Standardna devijacija
        mu (int/float): Mean/srednja vrednost
    """
    if case==1:
        #mu = 3.5
        #sigma = 0.5

        #count, bins, ignored = plt.hist(what, 30, density=True)
        #plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi))
        *np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
            #linewidth=2, color='r')
        what['zvezdice'].value_counts().sort_index().plot(kind="bar")
        mean = round(what['zvezdice'].mean(),2)
        median = what['zvezdice'].median()
        plt.axvline(mean-what['zvezdice'].min()+0.5, color='r',
        linestyle='--',label='Srednja vrednost')
        plt.axvline(median-what['zvezdice'].min()+0.5, color='g',
        linestyle='--',label='Medijana')
        plt.legend(loc='upper right')

    elif case == 2:
        what['drzava'].value_counts().plot.bar()

        #"Raspodela zvezdica hotela"
        #"Broj zvezdica"
        #"Kolicina"
    elif case ==3 :
        what['kontinent'].value_counts().plot.bar()
```

```
    elif case ==4 :
        what['br_soba'].plot(kind = 'bar',xticks=[])
        mean = round(what['br_soba'].mean())
        median = what['br_soba'].median()
        plt.axhline(mean, color='r', linestyle='--',
        label='Srednja vrednost')
        plt.axhline(median, color='g', linestyle='--',
        label='Medijana')
        plt.legend(loc='upper right')


    plt.title(title)
    plt.xlabel(x)
    plt.ylabel(y)
    plt.savefig('pydata/pyplots/'+path)
    plt.cla()
    plt.clf()
```

Funkcija koja generiše grafike i čuva ih u formatu png u okviru foldera pyplots. Koristi matplotlib biblioteku. 4 use case-ova.

```
def paramValuesGenerator(s):
    return  ','.join(['%s' for x in range
    (len(s[s.find("(")+1:s.find(")")].split(",")))])

def trimPonude():
    #aranzman(aran_id,naziv,krece,vraca,nap,smestaj_id,p_id)
    aranzman = pd.read_csv(PATH_DATA+"aranzmani.csv")
    filteredCopy = pd.DataFrame()
    filteredCopy['naziv'] = aranzman['ime']
    filteredCopy['krece'] = aranzman['datum_pocetka']
    filteredCopy['vraca'] = aranzman['datum_zavrsetka']
    filteredCopy['smestaj_id'] = aranzman['smestaj_id']
    filteredCopy['p_id'] = aranzman['p_id']

    return filteredCopy
```

Ove dve funkcije su uslužne funkcije koje koristi naredna funkcija. Za zadatak imaju uređivanje DataFrame-ova kako bi kasnije mogli da se dodaju u bazu podataka.

```
def dataTrimming():
#zbog baze mora da se menja redosled kolona....

    def switchToKey(df,col,x):  #col ako nije ukazuje da se
    radi o dataframe else of series
```

```
    if col:
        ret = np.where(df[col]==x)
        return ret[0][0]+1

    return np.where(df==x)[0][0]+1

kontinenti = pd.read_csv(PATH_DATA+'gradovi.csv').kontinent.unique()   #DONE
drzave = pd.read_csv(PATH_DATA+'gradovi.csv').drop_duplicates(subset = 'drzava')
.drop(columns=['naziv'] ,axis=1)
drzave['kontinent']=drzave['kontinent'].apply(lambda x :
switchToKey(kontinenti,None,x))#DONE
drzave.reset_index(drop=True,inplace=True)
gradovi = pd.read_csv(PATH_DATA+'gradovi.csv').drop
(columns=['kontinent'],axis=1)
gradovi['drzava'] = gradovi['drzava'].apply(lambda x :
switchToKey(drzave,'drzava',x))             #DONE
hoteli = pd.read_csv(PATH_DATA+'hoteli.csv')
hoteli['grad'] = hoteli['grad'].apply(lambda x:
switchToKey(gradovi,'naziv',x))
sobe = pd.read_csv(PATH_DATA+"sobe.csv")
hoteli = hoteli.iloc[:,[0,4,3,2,1]]
sobe = sobe.iloc[:,[0,1,3,2]]
prevoz = pd.read_csv(PATH_DATA+'prevoz.csv')
np.set_printoptions(threshold=sys.maxsize)
#komb = np.empty([ceil(hoteli['br_soba'].sum()),2])
komb = np.array([0,0])
for i in range(1,len(hoteli.index)+1):
    k=1
    while k<=hoteli.loc[hoteli.index==i-1,"br_soba"].values[0]:
        komb=np.vstack([komb,[i,k%(len(sobe.index)+1) if
        k%(len(sobe.index)+1)!=0 else 1]])
        k+=1
komb = komb.astype(int)
kmb = pd.DataFrame(komb,columns=["h_id","s_id"])
kmb = kmb.loc[kmb.index!=0]
kmb=kmb.reset_index(drop=True)
kmb.to_csv(PATH_DATA+"combinations.csv",index=None)
kmb["h_id"],kmb["s_id"]  = kmb["h_id"].astype(str),
kmb["s_id"].astype(str)
rez = pd.read_csv(PATH_DATA+"rand_rez.csv")
aranzman = trimPonude()
aktivnosti = pd.read_csv(PATH_DATA+"aktivnosti.csv")
akt_u_gradu = pd.read_csv(PATH_DATA+"aktivnosti_u_gradu.csv")
akt_u_gradu['g_id'],akt_u_gradu['akt_id'],akt_u_gradu['smestaj_id']
= akt_u_gradu['g_id'].astype(str),akt_u_gradu['akt_id']
.astype(str),akt_u_gradu['smestaj_id'].astype(str)
```

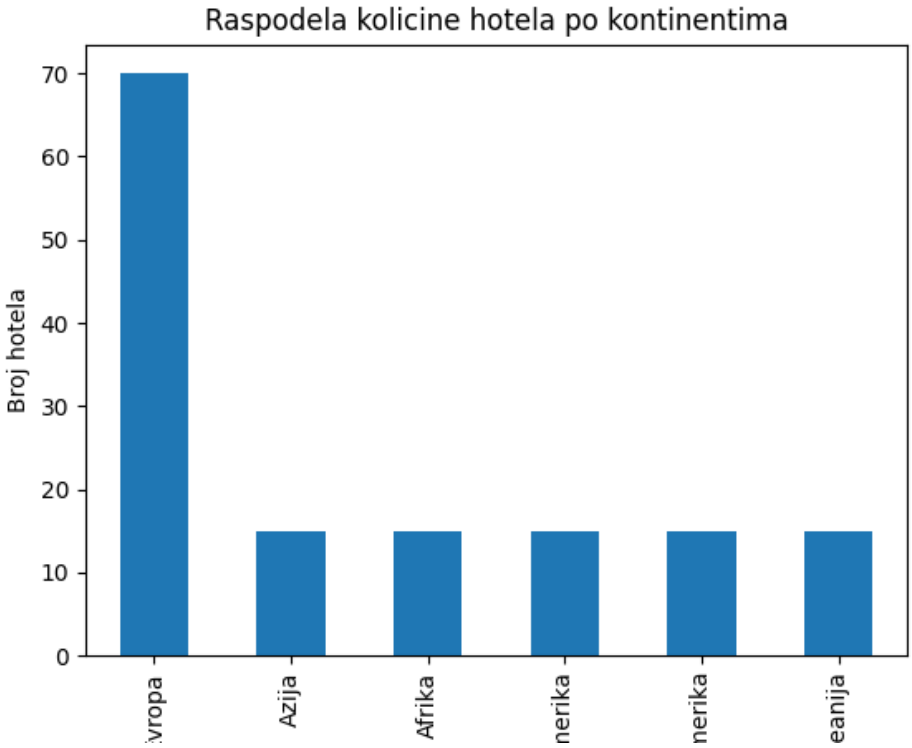
```
    akt_aran = pd.read_csv(PATH_DATA+"ima_aktivnost.csv")
    akt_aran['aran_id'],akt_aran['akt_id'] = akt_aran['aran_id'].
    astype(str),akt_aran['akt_id']
    .astype(str)
    return kontinenti,drzave,gradovi,hoteli,sobe,prevoz,
    kmb,aranzman,aktivnosti,akt_u_gradu,akt_aran,rez


def downloadCityImages(items):

    for x in items:
        x=x+"_city_center"
        downloader.download('_'.join([y for y in x.split(' ')]),
        limit=1, output_dir='slikeGradova', adult_filter_off=True,
        force_replace=False, timeout=90, verbose=True,filter="photo")
```

DataTrimming funkcija predstavlja najvažniju funkciju u okviru ove biblioteke. Ona uređuje DataFrame-ove za unos u bazu podataka. Optimizacija tipova podataka, rasporeda kolona, izbor kolona, omogućavanje unosa i DataFrame i numpy arraz tipova podataka u bazu podataka su neki od zadataka ove funkcije. Funkcija downloadCityImages koristi funckiju biblioteke bing.download.images kako bi pretražila i eventualno preuzela slike vezane za imena gradova. Fine-Tuning za query pretrage kako bi rezultati bili tačniji.

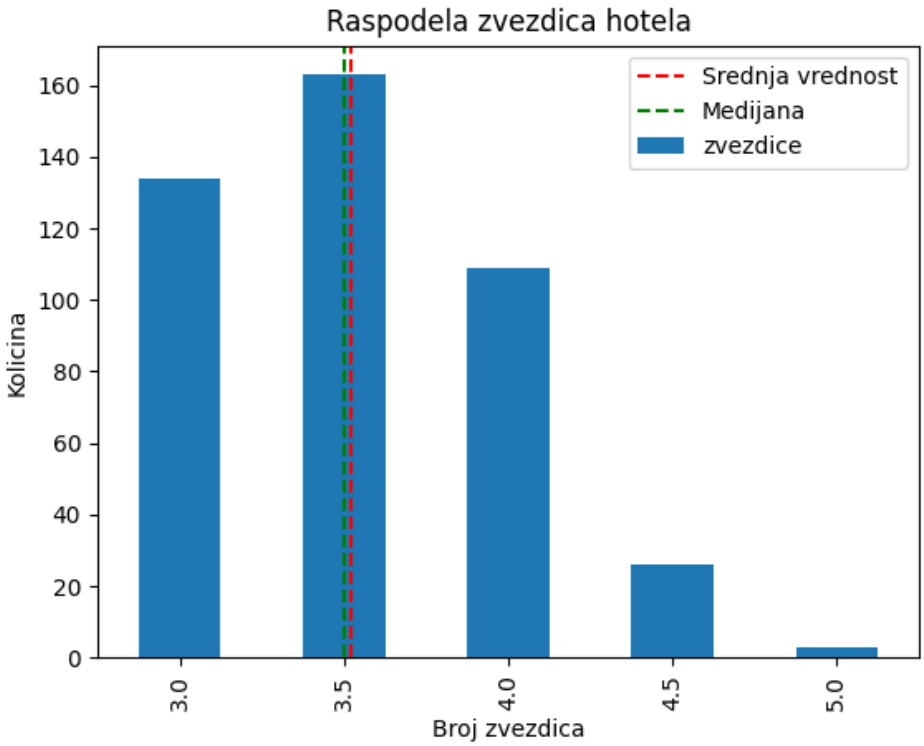
3.3 Zaključak sa graficima

U toku procesa generisanja podataka napravljeni su grafici kako bi se vizuelno prikazao rezultat.

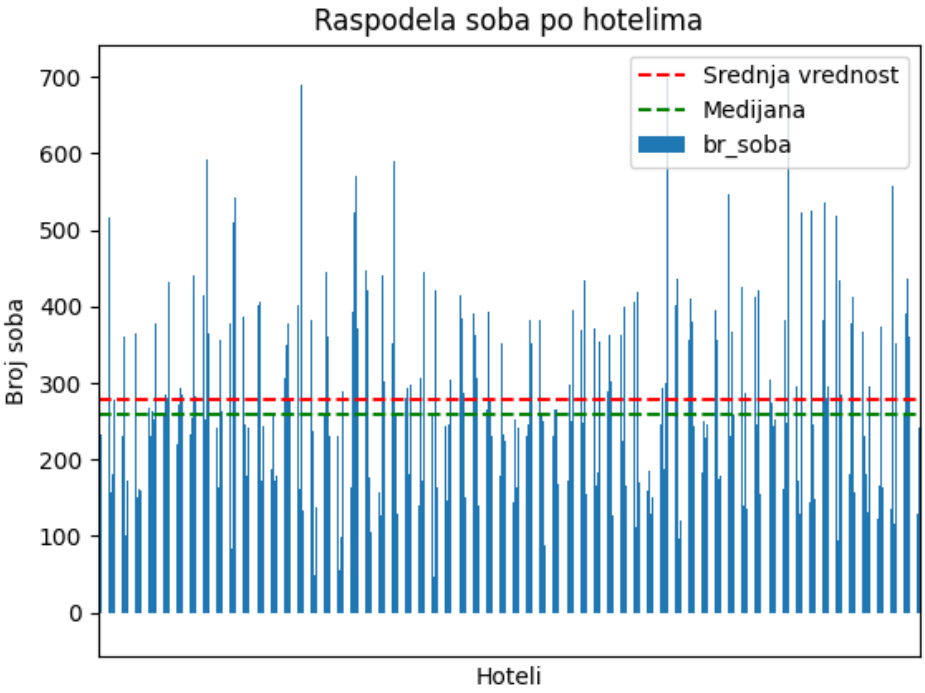
Slika 8: Raspodela hotela po kontinentima-fiktivna agencija se nalazi u Srbiji, tako da je optimalno staviti najveći broj hotela u Evropi.



Slika 9: Raspodela zvezdica hotela prati normalnu raspodelu



Slika 10: Raspodela broja soba po hotelima prati definisanu strukturu, i nema anomalija.



Prikaz podataka korišćenjem jupyter noteebook-a

February 14, 2023

[6]:
```
import pandas as pd

aran = pd.read_csv("aranzmani.csv")
smestaj = pd.read_csv("hoteli.csv")
akt_u_gradu = pd.read_csv("aktivnosti_u_gradu.csv")
aktivnosti = pd.read_csv("aktivnosti.csv")
gradovi = pd.read_csv("gradovi.csv")
ima_aktivnost = pd.read_csv("ima_aktivnost.csv")
prevoz = pd.read_csv("prevoz.csv")
sobe = pd.read_csv("sobe.csv")
hotel_to_room = pd.read_csv("combinations.csv")
```

[7]:
```
aran.head(1)
```

[7]:
```
              naziv    grad  zvezdice  br_soba             adresa       tip  \
0  ShangriLa Palace  Šangaj       3.5      301  7851 Wolfe Harbor  airplane

         tip_komp  p_id   prevod  broj_dana  mesec     mpr  smestaj_id  \
0  Pastel Voyages     1  avionom          3      1  Januar           1

         datum_pocetka      datum_zavrsetka  godina               m_str  \
0  2023-01-13 10:00:00  2023-01-16 10:00:00    2023  2023-01-13 10:00:00

                                          ime
0  Šangaj Januar 2023 ShangriLa Palace avionom 3 ...
```

[8]:
```
smestaj.head(1)
```

[8]:
```
              naziv    grad  zvezdice  br_soba             adresa
0  ShangriLa Palace  Šangaj       3.5      301  7851 Wolfe Harbor
```

[9]:
```
akt_u_gradu.head(1)
```

[9]:
```
   g_id  akt_id  smestaj_id
0    48       5           1
```

[10]:
```
aktivnosti.head(1)
```

[10]:
```
             naziv
0  Setnja po gradu
```

[11]:
```
gradovi.head(1)
```

[11]:
```
    naziv drzava kontinent
0  Šangaj   Kina     Azija
```

[12]:
```
ima_aktivnost.head(1)
```

[12]:
```
   aran_id  akt_id
0        1       5
```

[13]:
```
prevoz.head(1)
```

[13]:
```
        tip tip_komp  cena
0  airplane  Tour Go  1849
```

[14]:
```
sobe.head(1)
```

[14]:
```
     tip  br_kreveta                                              opis  \
0  SUITE           1  Soba u hotelu (dve prostorije koje ne moraju b...

   gen_cena
0      17.5
```

[15]:
```
hotel_to_room.head(1)
```

[15]:
```
   h_id  s_id
0     1     1
```

[ ]:
```

```