

**VISOKA TEHNIČKA ŠKOLA
STRUKOVNIH STUDIJA
S U B O T I C A**

PROJEKAT
iz predmeta Elektronsko poslovanje

KANDIDAT

Boris Evetović 16119219

Stevan Kazi 16119234

MENTOR

dr Zlatko Čović

SUBOTICA 2022. god.

SADRŽAJ

OPIS ZADATKA	1
Web sajt - turističke destinacije	1
Google Analytics - Tatto-sides	2
REALIZACIJA ZDATKA	3
Admin - dodavanje destinacije	3
Admin - dodavanje kategorije	4
Registracija	4
Pretraga destinacija	5
Dodavanje destinacije u liste	7
Like destinacije	8
Komentarisanje destinacije	10
Google Analytics	10
OPIS FUNKCIONALNOSTI	12
Tourism	12
Korisnik	12
Admin	15
Google analytics - tatto-side	18
KORIŠĆENA LITERATURA	20

OPIS ZADATKA

Web sajt - turističke destinacije

Ideja informacionog sistema je da se kroz aplikaciju, na što jednostavniji način vrši pregled i pretraga velikog broja dostupnih turističkih destinacija. Sam pristup web aplikaciji je omogućen samo registrovanim korisnicima, a registracija se vrši kroz vrlo jednostavan registracioni formular i verifikaciju email-a klikom na verifikacioni link koji se dostavlja putem email-a nakon popunjavanja formulara. Sama aplikacija razlikuje dve vrste korisnika: admin korisnike i “obične” korisnike, odnosno ciljnu grupu.

Administracija je dostupna samo admin korisnicima (u daljem tekstu “admin”) i putem njenog interfejsa može da se vrše različite manipulacije nad samim destinacijama i entitetima vezanim za destinaciju. Admin može da dodaje nove destinacije, definišući osnovne informacije kao što su naziv, adresa, geografski položaj (latituda i longituda - potrebno za kasniji prikaz na mapama i potencijalnu pretragu), opis, kategoriju i grad u kojoj se nalazi određena destinacija. Lista gradova je unapred definisana (gradovi Srbije), dok se kategorije takođe definišu putem administracije od strane admina. Kategorija ima svoj naziv i ikonicu koja smisleno simboliše kategoriju, a koristi se za kasniji prikaz na interfejsu aplikacije (ikonice su unapred definisane). Takođe, mogu da se vrše i izmene podataka postojećih destinacija i kategorija. Moguće je i obrisati određenu destinaciju ili kategoriju, s tim što, u sličaju brisanja kategorije, treba voditi računa da nijedna destinacija nije povezana za kategoriju koja se briše. Brisanje destinacije može da se uradi “bezbrizno”. Pored osnovnih (za funkcionalnost aplikacije) informacija o destinacijama, potrebno je dodati i slike pojedinačnih destinacija, u svrhu vernijeg vizuelnog prikaza destinacije.

Na samom početku, registrovani korisnik (u daljem tekstu “korisnik”) se nalazi na početnoj stranici na kojoj se prikazuju: neke od najpopularnijih destinacija u Srbiji (recimo “top 10”), neke od najposećenijih destinacija u Srbiji (recimo “top 10”), najpopularnije ili najposećenije (ili oba) destinacije u njegovom gradu (koji je prvobitno definisan putem registracionog formulara, sa kasnijom mogućnošću izmene) Nakon toga, korisnik, putem menija sa leve strane aplikacije (side menu), može da navodi interakciju sa aplikacijom ka raznim stranicama. Jedna od najbitnijih stranica jeste pretraga, gde korisnik može da vrši pretragu destinacija definišući sledeće karakteristike destinacije (u međusobnom odnosu ne “isključive”, odnosno višestruka pretraga) : naziv destinacije (delimično poklapanje), grad ili više gradova, kategorija ili više kategorija.

U pretragu su takođe uključene i opcije sortiranja po popularnosti i po posećenosti destinacija, u rastućem ili opadajućem redosledu. Na kraju, u pretrazi postoji još jedan

filter (“u blizini” ili “u mom gradu”) pomoću kojeg se na prethodnu pretragu dodaje uslov da prikazane destinacije moraju biti iz grada iz kojeg je korisnik. Ova opcija isključuje opciju (polje) pretrage po gradu ili po više gradova. Svaka destinacija koja je iz istog grada iz kojeg je i korisnik, prilikom samog prikaza na interfejsu nosi markicu (zastavicu, en. flag) “u blizini”.

Prikaz destinacija kroz pretragu, odnosno bilo koji višestruki prikaz destinacija je u kraćem obliku, odnosno prikazuju se neke osnovne informacije o destinaciji, dok se sve informacije prikazuju ulaskom na samu stranicu pojedinačne destinacije. Na stranici pojedinačne destinacije, korisnik ima mogućnost, pored detaljnog pregleda informacija, dodavanje review-a (komentara) na destinaciju (neograničeno) i “lajkovanja” ili “dislajkovanja” destinacije (ukoliko nije “lajkovana” destinacija prikazuje se like opcija, a ukoliko je “lajkovana” prikazuje se dislike opcija).

Korisnik ima mogućnost dodavanja destinacija u liste “omiljeno” i “za posetiti”. To može uraditi na stranici prikaza pojedinačne destinacije, a takođe ima i opciju da izbacuje određene destinacije iz listi. Liste i imaju i posebnu stranicu za prikaz.

Korisnik ima i opciju jednostavnog podešavanja profila, jer nije potreban veliki broj korisničkih informacija za funkcionalnost aplikacije. Korisnik može da menja ime, prezime i grad u kom živi. Dodatno može da promeni predefinisano sliku/avatara koju nakon registracije ima svaki korisnik.

Google Analytics - Tatto-sides

Google analytics projekat se odnosi na podešavanje google analitike i seo optimizaciju web prezentacije Tatto-sides koja se odnosi na tetovaže. Na web prezentaciju je uvezan google tag manager, preko kojeg je omogućeno podešavanje google analitike. Pomoću analitike se prati ponašanje korisnika na sajtu i u skladu sa time se prave potencijalne izmene i unapređenja. Seo optimizacija je zadovoljena različitim meta tagovima i ispraćenim pravilima i preporukama za omogućavanje boljeg indeksiranja sajta od strane google spider-a. Neki od najbitnijih meta tagova koji su definisani su description, keywords, canonical link, og tagovi. Takođe, struktura sadržaja na sajtu je struktuirana pomoću h tagova u skladu sa pravilima/predlozima seo optimizacije. Web prezentacija je dostupna i mobilnim uređajima što je takođe dodatno poboljšanje optimizacije. Postoje i linkovi ka “partnerskim” web prezentacijama u foot-eru sajta. Sajt je raspodeljen na različite stranice radi lakšeg praćenja analitike i boljeg google indeksiranja. Radi boljeg mesta prikaza u samoj pretrazi, lakšeg pamćenja i jednostavne pretrage, naziv sajta i parametri za pretragu treba da budu unikatni, što je više moguće.

REALIZACIJA ZDATKA

Projekat je rađen u Symfony PHP Framework-u (backend, php8.1) i React.js-u (frontend). U realizaciji zadatka su dati primeri najbitnijih funkcionalnosti (backend) web sajta.

Admin - dodavanje destinacije

Endpoint `/admin/destinations` se poziva POST request-om prilikom kreiranja destinacije, prosleđuju mu se određeni request json parametri.

```
##DestinationController.php

#[Route(path: '/admin/destinations', methods: ['POST'])]
public function create(Request $request): JsonResponse
{
    $destination = $this->destinationService->create(request: $request);

    if ($destination instanceof ErrorResponse) {
        return $this->json($destination, Response::HTTP_BAD_REQUEST);
    }

    return $this->jsonDestinationRead($destination, Response::HTTP_CREATED);
}
```

Funkcija *create* koja se nalazi u *destinationService* klasi pokušava da deserijalizuje request body u *Destination* entitet, a zatim na sličan način pokušava da ekstrahuje grad i kategoriju iz request-a i dodeli ih destinaciji (pravljenje relacije).

```
##DestinationService.php

public function create(Request $request): ErrorResponse|Destination
{
    $destination = $this->crud->deserializeEntity(request: $request, entityClass:
Destination::class);

    if ($destination instanceof ErrorResponse) {
        return $destination;
    }

    if ($destination instanceof Destination) {
        $city = $this->crud->extractCityFromRequest(request: $request);
        $category = $this->crud->extractCategoryFromRequest(request: $request);

        if ($city instanceof ErrorResponse) {
            return $city;
        }

        if ($category instanceof ErrorResponse) {
            return $category;
        }

        $destination->setCity($city)->setCategory($category);

        $this->crud->create($destination);
        return $destination;
    }

    return new ErrorResponse(message: 'Something went wrong');
}
```

Admin - dodavanje kategorije

Dodavanje kategorije funkcioniše na sličan princip kao i dodavanje destinacije. Funkcija *create* u klasi *CategoryService.php* pokušava da deserijalizuje request body u *Category* entitet i upiše ga u bazu.

```
##CategoryController.php

#[Route(path: '/admin/categories', methods: ['POST'])]
public function create(Request $request): JsonResponse
{
    $category = $this->categoryService->create(request: $request);

    if ($category instanceof ErrorResponse) {
        return $this->json($category, Response::HTTP_BAD_REQUEST);
    }

    return $this->jsonCategoryRead($category, Response::HTTP_CREATED);
}
```

```
##CategoryService.php

public function create(Request $request): ErrorResponse|Category
{
    $category = $this->crud->deserializeEntity(request: $request, entityClass:
Category::class);

    if ($category instanceof ErrorResponse) {
        return $category;
    }

    if ($category instanceof Category) {
        $this->crud->create($category);

        return $category;
    }

    return new ErrorResponse(message: 'Something went wrong');
}
```

Registracija

Endpoint */api/register* prima POST request sa parametrima u request body-ju (email, firstname, lastname...) i pomoću funkcije *userFromRequest* u klasi *AuthenticationService.php* pokušava da kreira User entitet pomoću datih parametara. Zatim pokušava User entitetu da doda i relaciju ka gradu u kome živi, kojeg je definisao takođe pomoću registracione forme. Zatim se radi validacija User entiteta (da li je sve popunjeno i da li su polja validna).

```
##AuthController.php

#[Route(path: '/api/register', methods: ['POST'])]
public function register(Request $request): JsonResponse
{
    $user = $this->authenticationService->userFromRequest(request: $request, groups:
[User::GROUP_REGISTER]);

    if (!$user) {
        return $this->json(new ErrorResponse(
            message: 'Server Error',
        ), Response::HTTP_INTERNAL_SERVER_ERROR);
    }

    $user = $this->authenticationService->bindCity(request: $request, user: $user);

    $violations = $this->authenticationService->validateUser(user: $user, groups:
```

```
[User::GROUP_REGISTER]);

    if (count($violations) > 0) {
        return $this->json(new ErrorResponse(
            message: 'Registration Error',
            errors: AuthenticationService::formatViolations($violations)
        ), Response::HTTP_BAD_REQUEST);
    }

    $user = $this->authenticationService->registerUser($user);

    $this->listService->prependUserLists(user: $user);

    return $this->jsonUserRead($user);
}
```

Ukoliko je sve uredi, poziva se funkcija *registerUser* u klasi *AuthenticationService.php* koja korisniku dodeljuje rolu, verifikacione parametre, profilnu sliku i hešovanu lozinku. Takođe, asinhrono šalje i verifikacioni email (dispatch *UserRegistered*). Na kraju upisuje korisnika u bazu.

```
##AuthenticationService.php

public function registerUser(User $user): User
{
    $user
        ->setVerificationToken(self::verificationCode())
        ->setVerificationTokenExpire((new \DateTime('now'))->modify('+5 minutes'))
        ->setRoles([User::ROLE_USER])
        ->setAvatar(User::DEFAULT_AVATAR);

    $user->setPassword($this->hasher->hashPassword($user, $user->getPlainPassword()));

    $user->eraseCredentials();

    $this->messageBus->dispatch(new UserRegistered(
        email: $user->getEmail(),
        verificationCode: $user->getVerificationToken()
    ));

    $this->entityManager->persist($user);
    $this->entityManager->flush();

    return $user;
}
```

Pretraga destinacija

Endpoint */api/destinations/by* prima POST request sa parametrima za pretragu destinacije.

```
##DestinationController.php

#[Route(path: '/api/destinations/by', methods: ['POST'])]
public function listBy(Request $request): JsonResponse
{
    $response = $this->destinationService->listByCriteria(request: $request);

    if ($response instanceof ErrorResponse) {
        return $this->json($response, Response::HTTP_INTERNAL_SERVER_ERROR);
    }

    return $this->json($response);
}
```

Funkcija *listByCriteria* pokušava da ekstraktuje parametre za pretragu iz request body-ja. Unesu su svi dati parametri u niz kriterija i niz se prosledi funkciji *searchByCriteria* u klasi *DestinationRepository.php*.

```

##DestinationService.php

public function listByCriteria(Request $request): ErrorResponse|array
{
    try {
        $params = json_decode((string)$request->getContent(), false, 512,
JSON_THROW_ON_ERROR);
    } catch (\JsonException $e) {
        return new ErrorResponse(message: 'List failed', errors: ['server' => $e-
>getMessage()]);
    }

    $criteria = [];
    $criteria['cityId'] = $params->cityId ?? null;
    $criteria['categoryId'] = $params->categoryId ?? null;
    $criteria['name'] = $params->name ?? null;
    $criteria['sort'] = $params->sort ?? null;
    $criteria['limit'] = $params->limit ?? null;
    $criteria['page'] = $params->page ?? null;
    $criteria['nearMe'] = $params->nearMe ?? null;

    if ($criteria['nearMe'] === true) {
        /** @var User $user */
        $user = $this->security->getUser();

        if ($user && $user->getCity()) {
            $criteria['cityId'] = $user->getCity()->getId();
        }
    }

    unset($criteria['nearMe']);

    return $this->destinationRepository->searchByCriteria(criteria: $criteria);
}

```

Funkcija *searchByCriteria* pravi sql query sa potrebnim parametrima.

```

##DestinationRepository

public function searchByCriteria(array $criteria): array
{
    $builder = $this->createQueryBuilder('d');

    if ($criteria['cityId'] !== null) {
        if (is_array($criteria['cityId'])) {
            $builder->andWhere('d.city IN (:cityId)')->setParameter('cityId',
$criteria['cityId']);
        } else {
            $builder->andWhere('d.city = :cityId')->setParameter('cityId',
$criteria['cityId']);
        }
    }

    if ($criteria['categoryId'] !== null) {
        if (is_array($criteria['categoryId'])) {
            $builder->andWhere('d.category IN (:categoryId)')->
setParameter('categoryId', $criteria['categoryId']);
        } else {
            $builder->andWhere('d.category = :categoryId')->setParameter('categoryId',
$criteria['categoryId']);
        }
    }

    if ($criteria['name'] !== null) {
        $builder->andWhere('d.name LIKE :name')->setParameter('name', '%' .
$criteria['name'] . '%');
    }

    $builder->orderBy('d.id');

    if (in_array($criteria['sort'], ['popularity', 'attendance'], true)) {
        if ($criteria['sort'] === 'popularity') {
            $builder->orderBy('d.popularity', 'DESC');
        }

        if ($criteria['sort'] === 'attendance') {
            $builder->orderBy('d.attendance', 'DESC');
        }
    }
}

```



```

    }
}

$limit = 10;
if ($criteria['limit'] && is_int($criteria['limit']) && $criteria['limit'] > 0) {
    $limit = $criteria['limit'];
}

$page = 1;
if ($criteria['page'] && is_int($criteria['page']) && $criteria['page'] > 0) {
    $page = $criteria['page'];
}

$builder->setFirstResult(($page - 1) * $limit);
$builder->setMaxResults($limit);

$paginator = new Paginator($builder->getQuery());

$result['totalResults'] = $paginator->count();
$result['page'] = $page;
$result['pagesCount'] = ceil($result['totalResults'] / $limit);
$result['items'] = $paginator->getIterator()->toArrayCopy();

return $result;
}

```

Dodavanje destinacije u liste

Endpoint-i `/api/lists/destinations/{id}/to-visit` i `/api/lists/destinations/{id}/favorites` primaju GET request i pomoću id-ja destinacije i korisnika koji je napravio request, dodaje destinaciju u željenu listu (svakom korisniku se tokom uspešne registracije kreiraju dve liste, to-visit i favorites).

```

#[Route(path: '/api/lists/destinations/{id}/to-visit', methods: ['GET'])]
public function addDestinationToVisit(int $id, #[CurrentUser] User $user):
    JsonResponse
    {
        $destination = $this->destinationService->findById(id: $id);

        if (!$destination) {
            return $this->json(null);
        }

        $list = $this->listService->appendDestination(user: $user, destination:
        $destination, type: WishList::TO_VISIT);

        return $this->jsonListRead(wishList: $list, status: Response::HTTP_ACCEPTED);
    }

#[Route(path: '/api/lists/destinations/{id}/favorites', methods: ['GET'])]
public function addDestinationFavorites(int $id, #[CurrentUser] User $user):
    JsonResponse
    {
        $destination = $this->destinationService->findById(id: $id);

        if (!$destination) {
            return $this->json(null);
        }

        $list = $this->listService->appendDestination(user: $user, destination:
        $destination, type: WishList::FAVORITES);

        return $this->jsonListRead(wishList: $list, status: Response::HTTP_ACCEPTED);
    }

```

Funkcija *appendDestination* u klasi *ListService.php* pronalzi željenu listu na osnovu korisnika i tipa liste. Zatim proverava da li destinacija postoji u listi, ako postoji onda je ukloni sa liste, a ako ne postoji onda je dodaje u listu (jer se isti endpoint-i koriste i za

uklanjanje destinacije iz liste).

```
##ListService.php

public function appendDestination(User $user, Destination $destination, string $type):
ErrorResponse|WishList
{
    /** @var WishList $list */
    $list = $this->listRepository->findByUserAndType(user: $user, type: $type);

    if ($list->getDestinations()->contains($destination)) {
        $list->removeDestination($destination);
    } else {
        $list->addDestination(destination: $destination);
    }

    $this->crud->patch(entity: $list);
    $this->crud->getEntityManager()->refresh($list);

    return $list;
}
```

Like destinacije

Endpoint-i `/api/destinations/{id}/like` i `/api/destinations/{id}/unlike` primaju GET request i preko id-ja destinacije i trenutnog korisnika dodaju like ili dislike na destinaciju.

```
##DestinationController.php

#[Route(path: '/api/destinations/{id}/like', methods: ['GET'])]
public function likeDestination(int $id, #[CurrentUser] User $user): JsonResponse
{
    $destination = $this->destinationService->findById(id: $id);

    if (!$destination) {
        return $this->json(new ErrorResponse(
            message: 'Fetch failed',
            errors: ['destination', 'not found']
        ));
    }

    $this->destinationService->addLike(destination: $destination, user: $user);

    return $this->json($destination, Response::HTTP_CREATED);
}

#[Route(path: '/api/destinations/{id}/unlike', methods: ['GET'])]
public function unlikeDestination(int $id, #[CurrentUser] User $user): JsonResponse
{
    $destination = $this->destinationService->findById(id: $id);

    if (!$destination) {
        return $this->json(new ErrorResponse(
            message: 'Fetch failed',
            errors: ['destination', 'not found']
        ));
    }

    $this->destinationService->undoLike(
        destination: $destination,
        user: $user
    );

    return $this->jsonDestinationRead(destination: $destination, status:
Response::HTTP_CREATED);
}
```

Funkcije *addLike* i *undoLike* proveravaju da li je na destinaciju već dodat like ili dislike i da li treba da obrišu suprotnu akciju (ako je dodat like, a već postoji dislike,

obriši dislike i obrnuto). Recimo da je korisnik lajkovao destinaciju, ako u bazi već postoji zapis da je prethodno lajkovana destinacija, onda se proverava da li je flag na zapisu “deleted” i ako jeste, postavi flag na false, a ako nije, postavi ga na true. Ako ne postoji nikakav prethodni zapis za lajk ili dislajk destinacije, kreiraj novi.

```
##DestinationService.php

public function addLike(Destination $destination, User $user): DestinationLike
{
    /** @var DestinationLike[] $previousLike */
    $previousLike = $this->likeRepository->isLikedByUser(destination: $destination,
user: $user);
    /** @var DestinationLike[] $previousDislike */
    $previousDislike = $this->likeRepository->isDislikedByUser(destination:
$destination, user: $user);

    if (count($previousDislike)) {
        $previousDislike[0]->setDeleted(true);
        $this->crud->patch($previousDislike[0]);
    }

    if (count($previousLike)) {
        if ($previousLike[0]->isDeleted()) {
            $previousLike[0]->setDeleted(false);
        } else {
            $previousLike[0]->setDeleted(true);
        }

        $this->crud->patch($previousLike[0]);
        return $previousLike[0];
    }

    $destination->setPopularity($destination->getPopularity() + 1);
    $this->crud->patch(entity: $destination);

    $like = (new DestinationLike())
        ->setDestinationId($destination->getId())
        ->setUserId($user->getId())
        ->setCreatedAt(new \DateTime())
        ->setDeleted(false)
        ->setNegative(false);

    $this->crud->create(entity: $like);

    return $like;
}

public function undoLike(Destination $destination, User $user): DestinationLike
{
    /** @var DestinationLike[] $previousLike */
    $previousLike = $this->likeRepository->isLikedByUser(destination: $destination,
user: $user);
    /** @var DestinationLike[] $previousDislike */
    $previousDislike = $this->likeRepository->isDislikedByUser(destination:
$destination, user: $user);

    if (count($previousLike)) {
        $previousLike[0]->setDeleted(true);
        $this->crud->patch(entity: $previousLike[0]);
    }

    if (count($previousDislike)) {
        if ($previousDislike[0]->isDeleted()) {
            $previousDislike[0]->setDeleted(false);
        } else {
            $previousDislike[0]->setDeleted(true);
        }

        $this->crud->patch($previousDislike[0]);
        return $previousDislike[0];
    }

    $like = (new DestinationLike())
        ->setDestinationId($destination->getId())
        ->setUserId($user->getId())
```

```

->setCreatedAt(new \DateTime())
->setDeleted(false)
->setNegative(true);

$destination->setPopularity($destination->getPopularity() - 1);

$this->crud->create(entity: $like);
$this->crud->patch(entity: $destination);

return $like;
}

```

Komentarisanje destinacije

Endpoint `/api/comments` prima POST request sa parametrima za kreiranje komentara i id-jem destinacije. Zatim se request i trenutni korisnik prosleđuju funkciji `create` u klasi `CommentService.php`.

```

#[Route(path: '/api/comments', methods: ['POST'])]
public function create(Request $request, #[CurrentUser] User $user): JsonResponse
{
    $comment = $this->commentService->create(request: $request, user: $user);

    if ($comment instanceof ErrorResponse) {
        return $this->json($comment, Response::HTTP_NOT_ACCEPTABLE);
    }

    return $this->jsonCommentRead(comment: $comment, status: Response::HTTP_CREATED);
}

```

Funkcija pokušava da kreira Comment entitet od request parametara i pokušava da ekstraktuje destinaciju pomoću id-ja destinacije iz request body-ja. Ako je sve uredu, prave se relacije između komentara i destinacije i komentara i korisnika.

```

public function create(Request $request, User $user): ErrorResponse|DestinationComment
{
    /** @var DestinationComment $comment */
    $comment = $this->crud->createFromRequest(request: $request, entityClass:
DestinationComment::class);

    $destination = $this->crud->extractDestinationFromRequest(request: $request);

    if ($destination instanceof ErrorResponse) {
        return $destination;
    }

    $comment
        ->setDestination($destination)
        ->setUser($user);

    $violations = $this->crud->validateEntity(entity: $comment);

    if (count($violations) > 0) {
        return new ErrorResponse(
            message: 'Invalid entity',
            errors: Crud::formatViolations($violations)
        );
    }

    $this->crud->create(entity: $comment);

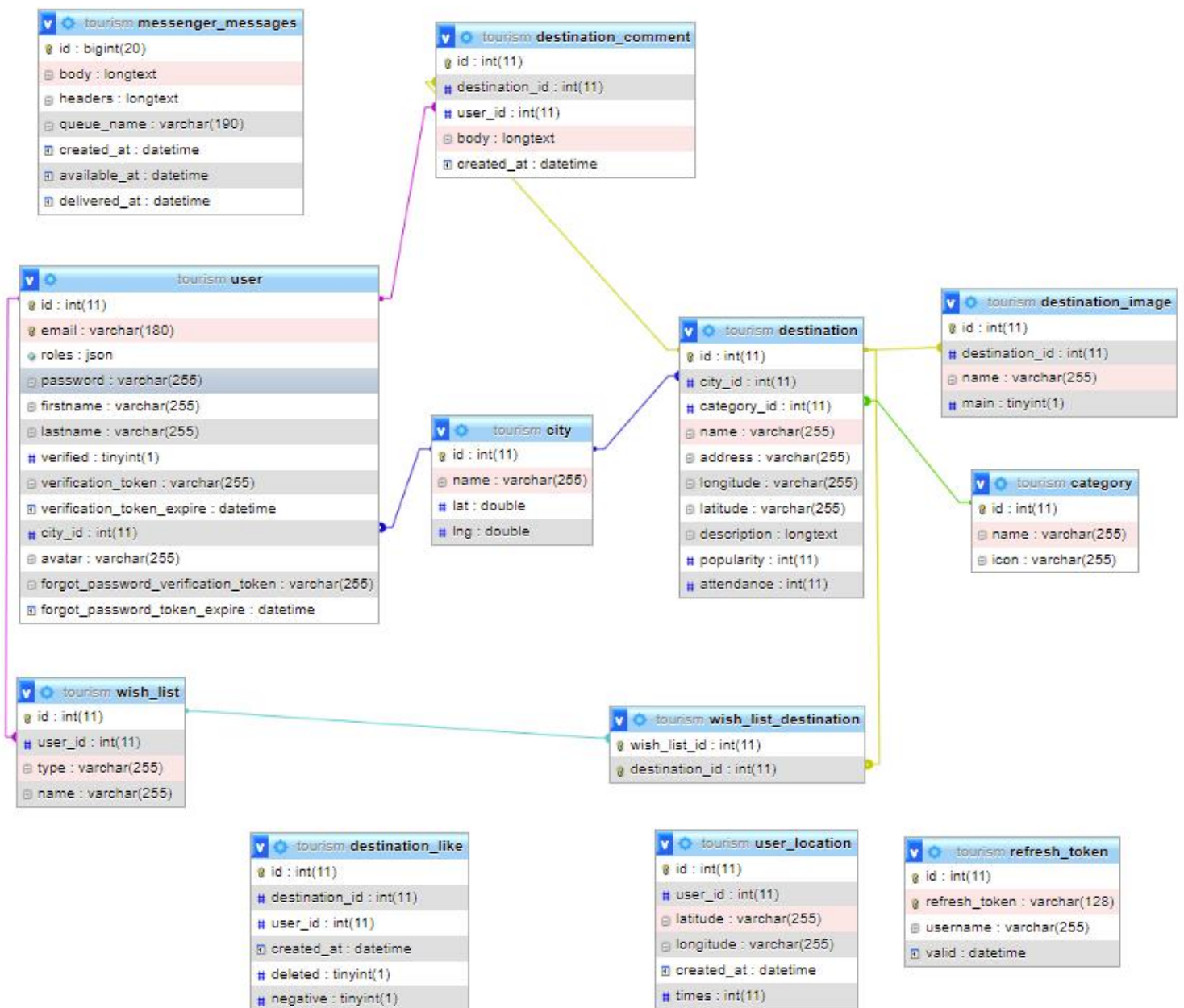
    return $comment;
}

```

Google Analytics

<https://autsajdrs.proj.vts.su.ac.rs/index.html>

STRUKTURA BAZE PODATAKA

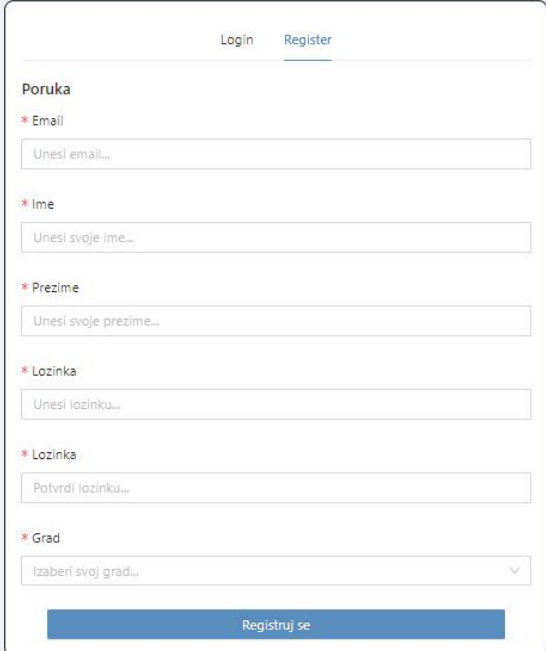


OPIS FUNKCIONALNOSTI

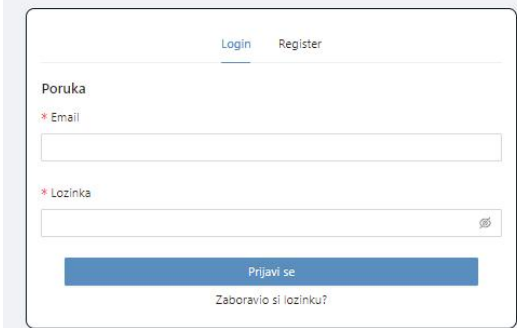
Tourism

Korisnik

Korisnik prvo mora da bude registrovan da bi imao pristup web sajtu. Prema tome prva funkcionalnost je registrationi formular. Korisnik popunjava registrationi formular navodeći neke osnovne informacije o sebi. Nakon toga mu stiže verifikacioni email sa verifikacionim linkom. Klikom na link korisnik verifikuje svoj email. Zatim može da se uloguje pomoću login formulara (email i šifra).



The registration form is titled "Register" and includes a "Login" link. It contains the following fields: "Email" (required), "Ime" (first name, required), "Prezime" (last name, required), "Lozinka" (password, required), "Lozinka" (confirm password, required), and "Grad" (city, required, dropdown menu). A "Registruj se" button is at the bottom.



The login form is titled "Login" and includes a "Register" link. It contains the following fields: "Email" (required) and "Lozinka" (password, required). A "Prijavi se" button is at the bottom, with a link "Zaboravio si lozinku?" below it.

Pretraga destinacija

Pretraga

Ime destinacije

test

Grad

Kragujevac

Kategorije

Kategorija

☐ Prikaži destinacije u blizini

Sortiraj po...

Pretraži

#Subotica test

#Subotica amin

destinacija - adresa 123

#Vajevro test

Test kroz UI - adr 1

#Subotica amin

Pretraga destinacija

Pretraga

Broj pronađenih lokacija: 4

test - test test

#Subotica test

destinacija - adresa 123

#Subotica amin

destinacija - adresa 123

#Vajevro test

Test kroz UI - adr 1

#Subotica amin

Na stranici za pretragu, korisnik može da popuni kriterijume za pretragu, a zatim će dobiti rezultate pretrage. Nakon toga, klikom na određenu destinaciju, korisnik može da ode na stranicu pojedinačne destinacije gde se vide sve dostupne informacije o destinaciji.

Stevan Kazi

test - Subotica

test test

1

0

Obrisi sa liste omiljenih

Obrisi sa liste za posetiti

Opis:

amin izmeni test

Stevan Kazi

1

0

Obrisi sa liste omiljenih

Obrisi sa liste za posetiti

Opis:

amin izmeni test

Komentari:

Stevan Kazi:

test komentar

7/3/2022

Unesi komentar

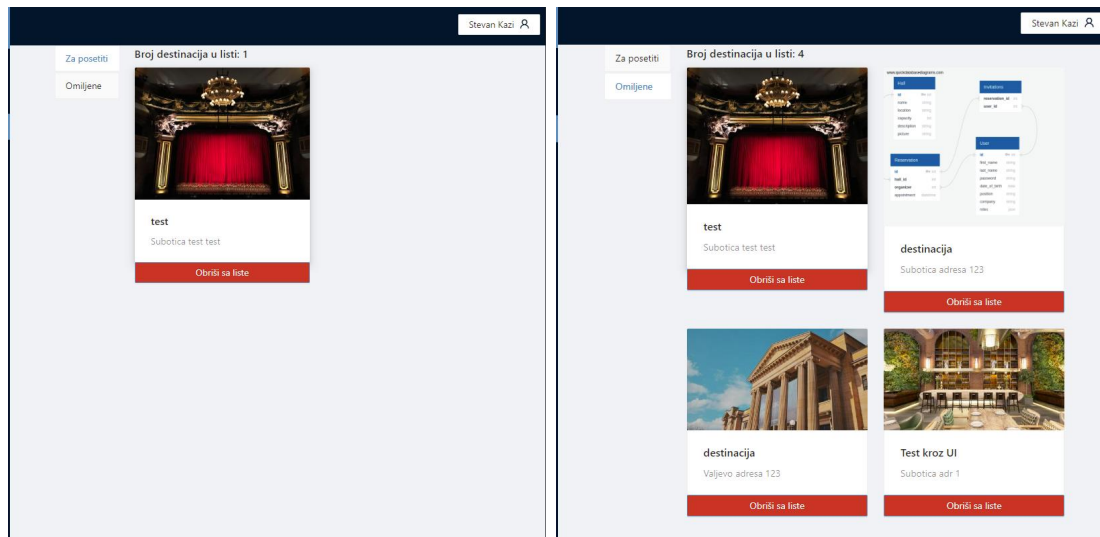
Dodaj komentar

Ant Design ©2018 Created by

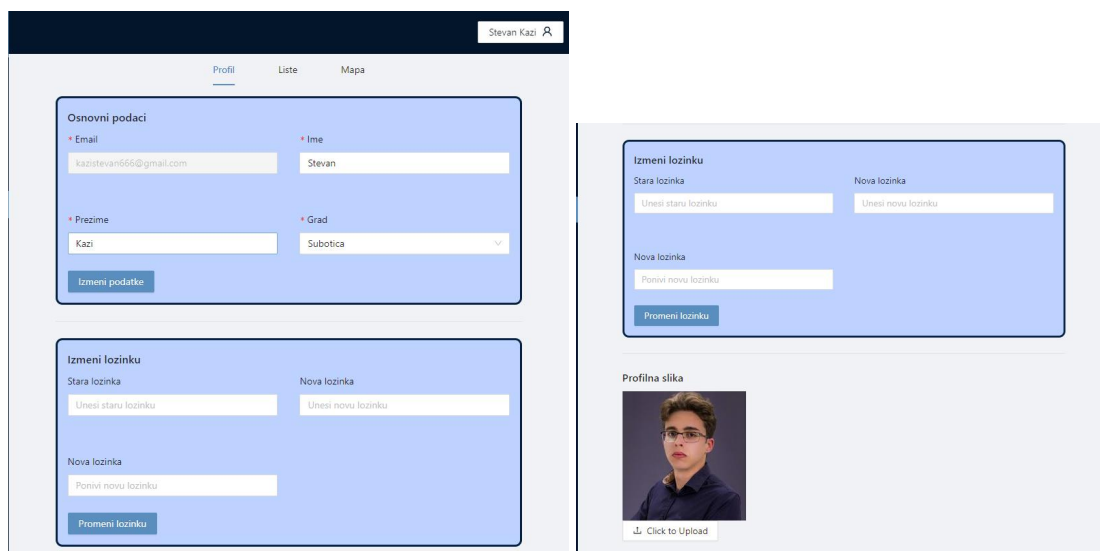
Ant UED

Na stranici pojedinačne destinacije, korisnik može da doda destinaciju u određenu listu, doda lajk ili dislajk, i doda komentar i pregleda komentare drugih korisnika.

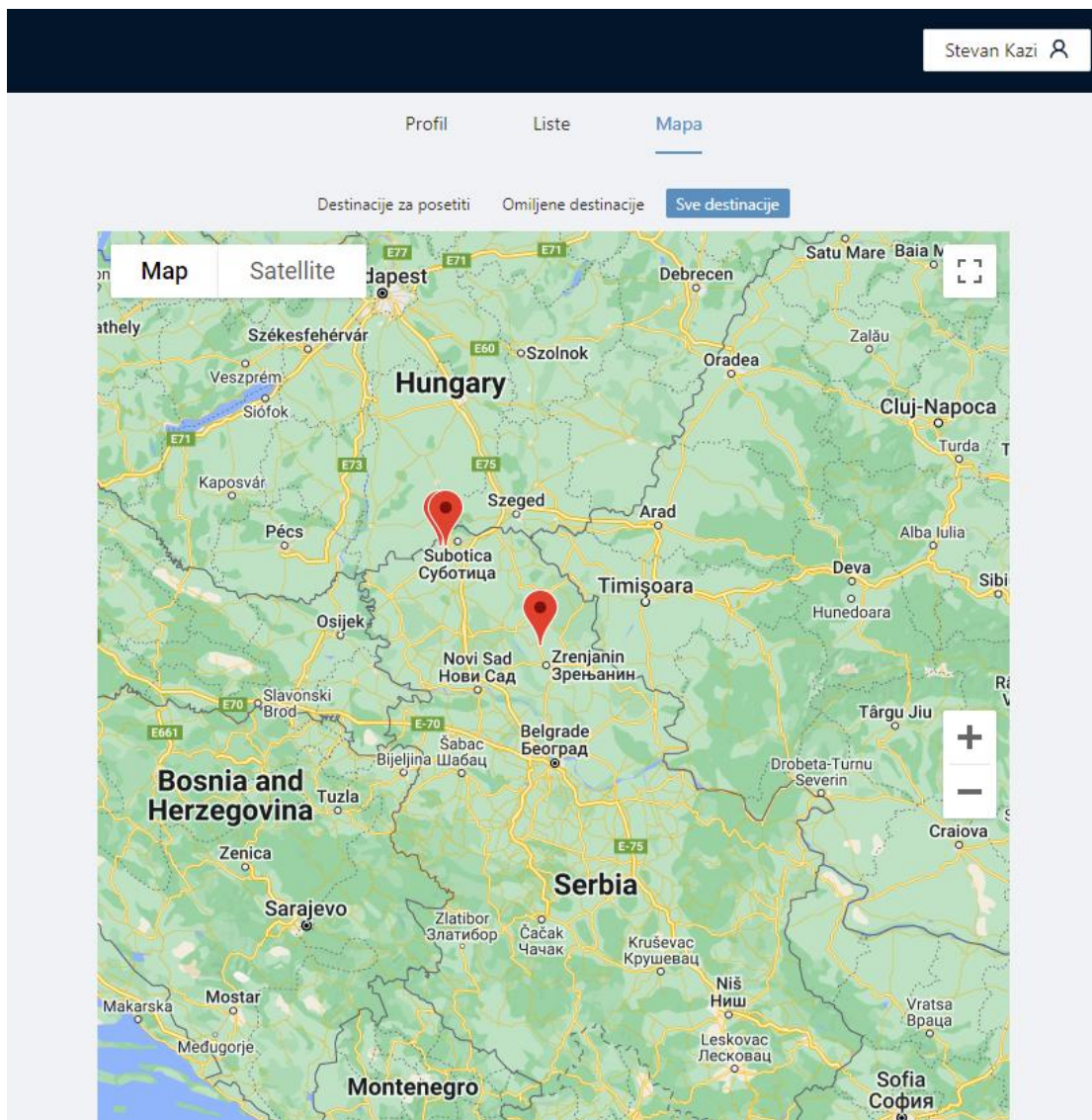
Zatim, postoji i stranica za korisnikove liste, gde korisnik može da pregleda sve destinacije koje je prethodno dodao u listu omiljenih ili za posetiti. Takođe može i da skine destinaciju sa liste.



Zatim idemo na profil korisnika. Korisnik u svom profilu može da menja neke osnovne informacije o sebi, da promeni šifru i da promeni profilnu sliku.



Takođe, može i da pristupi listama klikom na tab "liste". Na poslednjem tabu koji se zove mapa korisnik može da pregleda na mapi destinacije koje je dodao u listu omiljenih ili za posetiti.



Admin

Administrator ima pristup stranicama za dodavanje, editovanje i brisanje kategorija i destinacija.

Unesi novu destinaciju

Id	Ime destinacije	Adresa	Grad	Kategorija	Akcije
1	test	test test	Subotica	test	 
2	destinacija	adresa 123	Subotica	amin	 
3	destinacija	adresa 123	Valjevo	test	 
4	Test kroz UI	adr 1	Subotica	amin	 

* Ime destinacije

Unesi ime destinacije...

* Unesi adresu destinacije

Unesi adresu destinacije

Opis destinacije

Unesi opis...

* Kategorija

Izaberi kategoriju...

* Grad

Izaberi grad

Izaberi lokaciju



Cancel

OK

Stevan Kazi

Unesi novu kategoriju

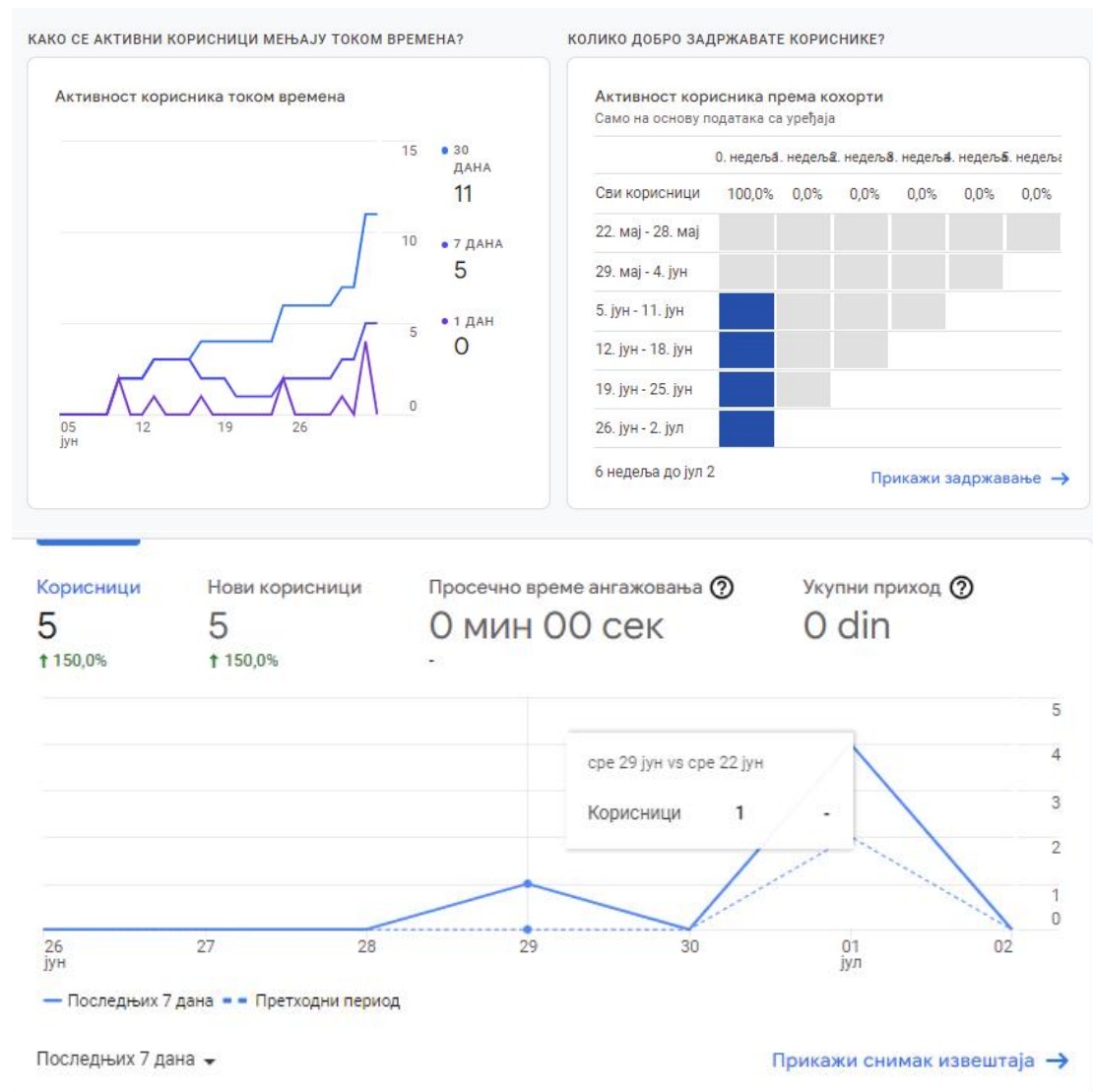
Id	Ikonica kategorije	Ime kategorije	Akcije
1		test	
2		amin	
3		test	

< 1 >

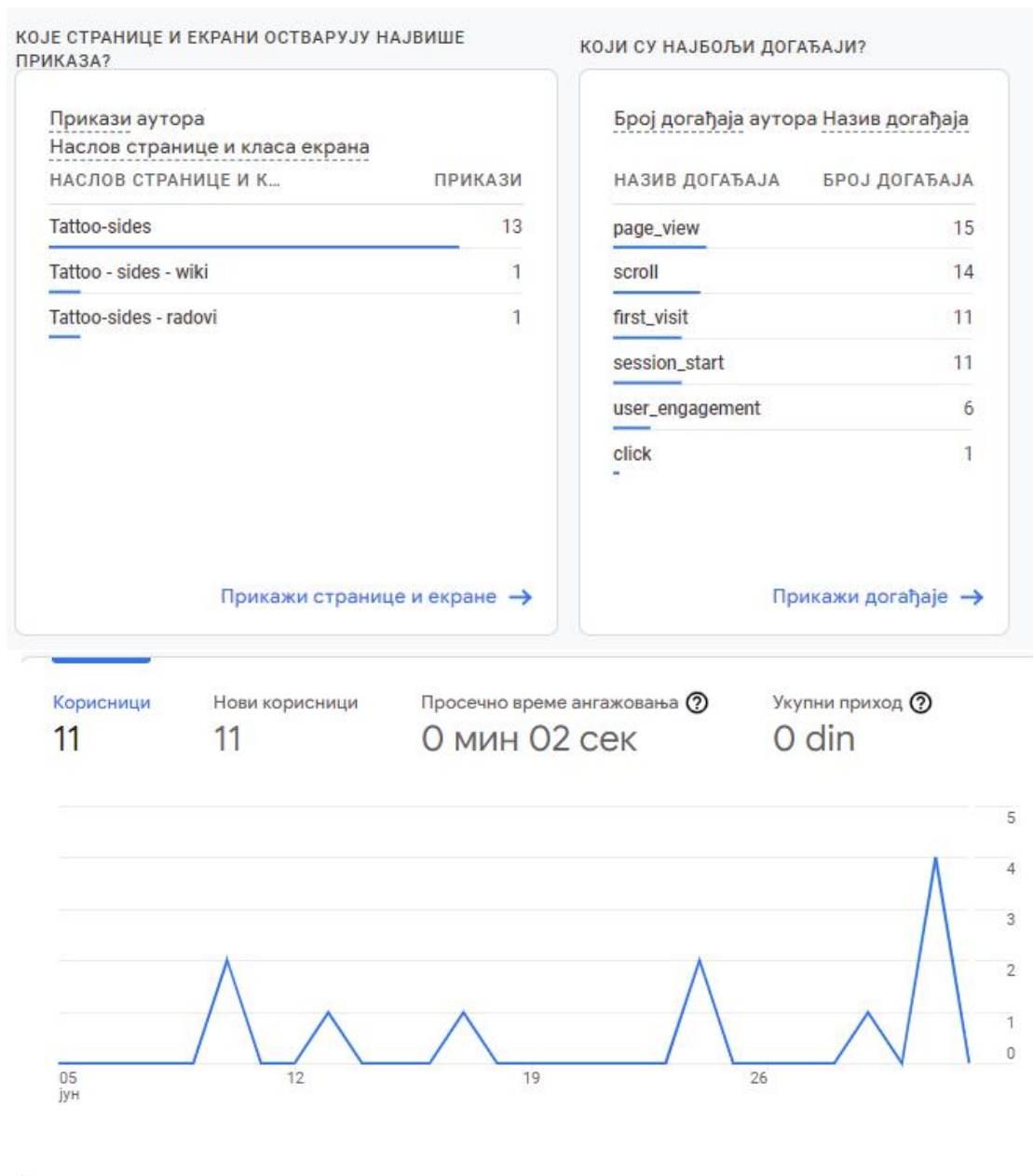
Google analytics - tatto-side

Tatto sides je web prezentacija vezana za tetovaže. Na ovom sajtu se prati korisničko ponašanje, kretanje i poseta sajtu kroz google analitiku.

Pomoću google analitike možemo da vidimo, u poslednjih 7 dana je broj korisnika drastično skočio u odnosu na period od samog nastanka sajta (skoro 50%).



Najposećenija stranica je početna stranica, a najbolji događaji na sajtu su pregled stranice i skrol.



KORIŠĆENA LITERATURA

- W1. <https://symfony.com/doc/current/index.html>
- W2. <https://reactjs.org/docs/getting-started.html>
- W3. <https://github.com/lexik/LexikJWTAuthenticationBundle>