

**Документација за проект по предметот Имплементација на  
софтверски системи со слободен и отворен код**



**Изработено од:**

Момчило Илиев, 183248

Сани Раданлиева, 203120

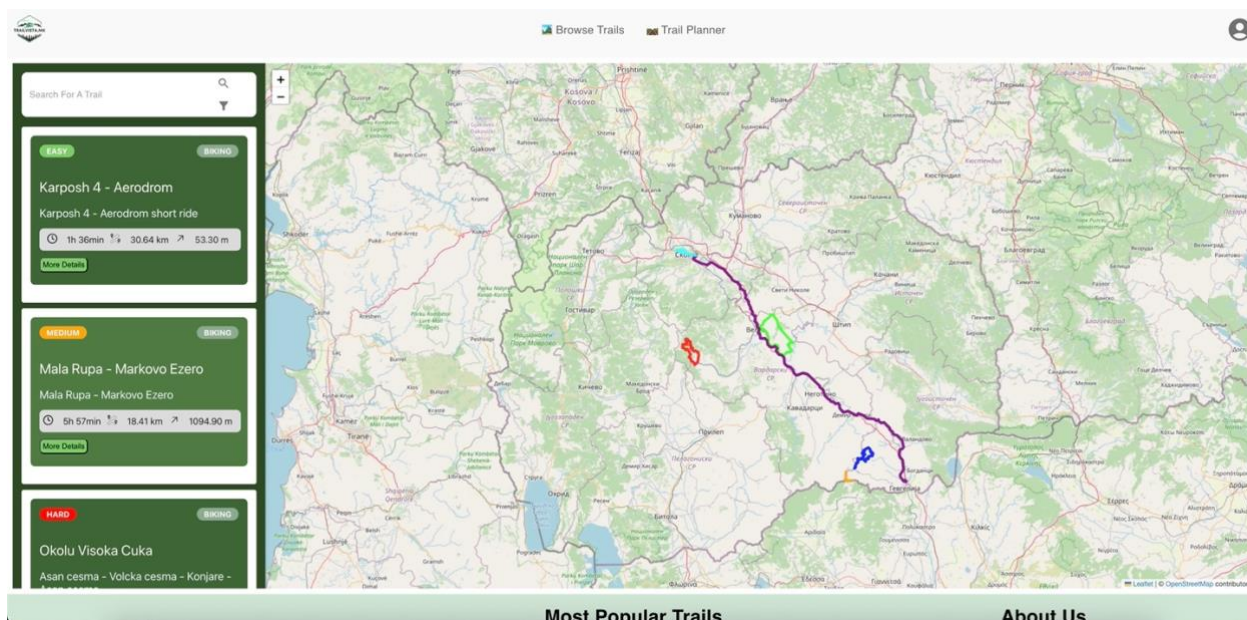
# 1. Вовед

**TrailVista.mk** е веб апликација за додавање и преглед на нови и веќе постоечки патеки за велосипедизам, планинарење и трчање. За креирање на оваа веб апликација користено е **Laravel** (за backend делот), **React** (за frontend делот), а за база на податоци користено е **PostgreSQL**. За функционалностите поврзани со прикажување и користење на мапи, користено е **OpenStreetMap**.

## 2. Основни функционалности

Основни функционалности на TrailVista.mk се:

- **Преглед и пребарување на рути** - корисниците можат да пребаруваат низ веќе постоечки рути според различни филтри, како тежина на рута, должина на рута, вид на рута, време на движење и слично, и истата патека да ја прегледаат детално

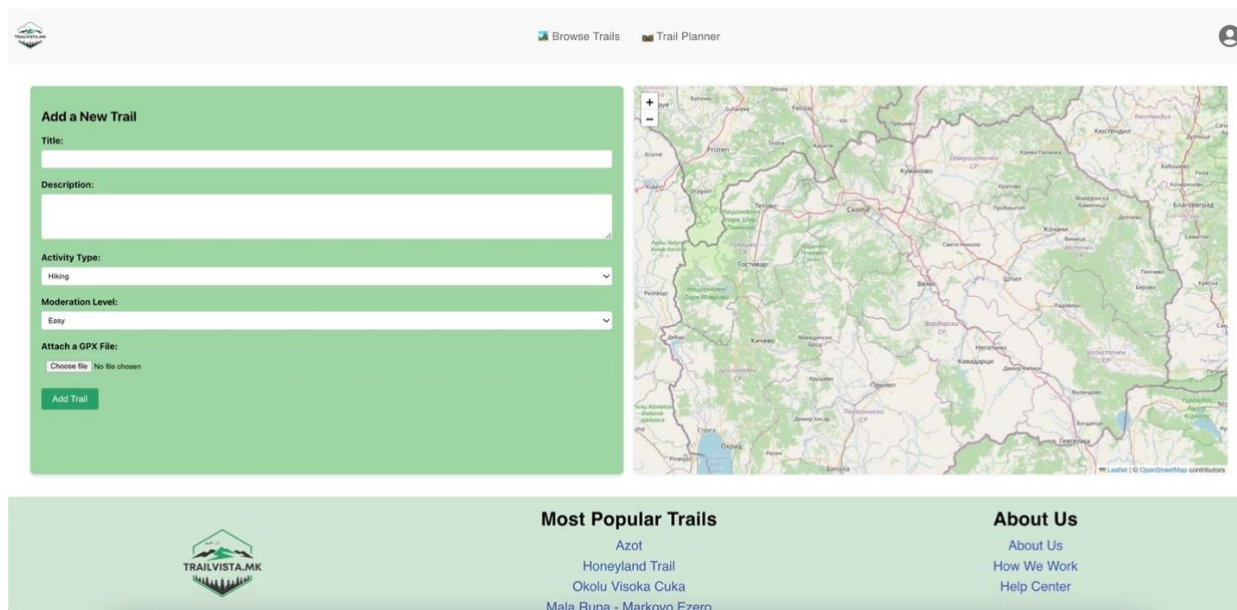


Слика 1. Преглед и пребарување на рути

Во рамки на оваа страна, прикажани се на мапата веќе постоечките рути. За функционалноста поврзана со мапата користена е **OpenStreetMap**, со библиотека **react-leaflet**, преку која се креира **MapContainer** во кој е прикажана мапата. Се користи **Polyline** (исто од библиотеката **react-leaflet**) за исцртување на рутите кои се прикажани на мапата.

На левата страна може да се види една **Sidebar** компонента во која се излистани сите постоечки рути. Овде може да се пребарува по името на рутата, или да се искористи филтерот за понапредно пребарување.

- **Додавање на нова рута** - корисниците можат да додадат нова рута, внесувајќи име на рута, краток опис, вид на активност/спорт и тежина на рутата, и грх фајл



The screenshot shows the 'Add a New Trail' form on the Trail Vista website. The form includes fields for Title, Description, Activity Type (Hiking), Moderation Level (Easy), and an option to Attach a GPX File. A map of North Macedonia is displayed on the right side of the form. Below the form, there is a section for 'Most Popular Trails' listing Azot, Honeyland Trail, Okolu Visoka Cuka, and Mala Rupa - Markovo Ezero. To the right of this section is an 'About Us' link.

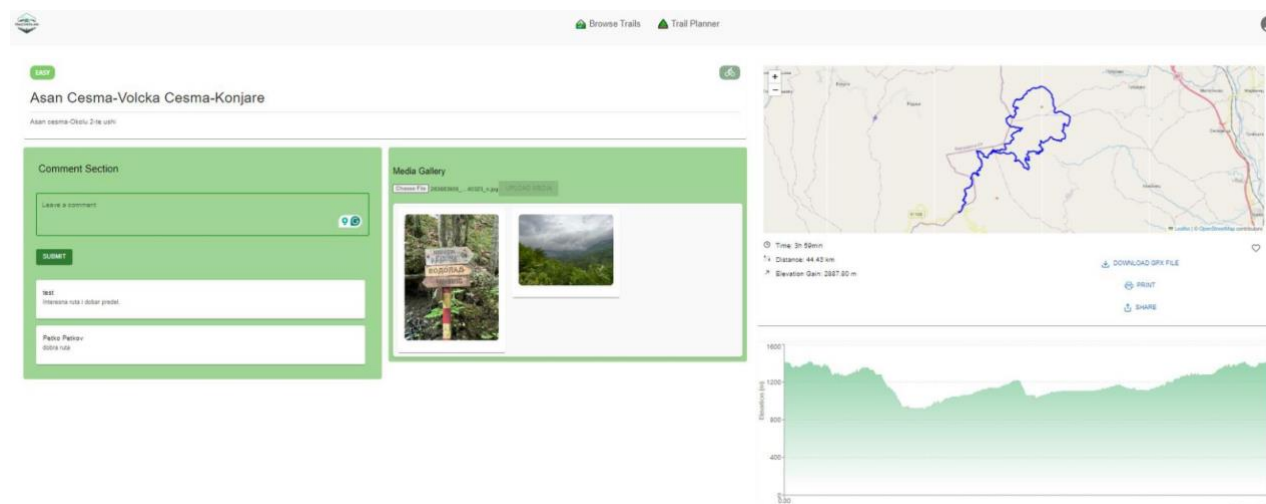
Слика 2. Додавање на нова рута

- **Користење на интерактивна мапа** - корисниците можат да ја искористат интерактивната мапа за да ја видат целосната патека, нејзината должината, тежината и мапа со елевација на целата рута

- **Оставање коментари за рутата** - корисниците можат да ги оценуваат рутите, споделувајќи искуства, фотографии и видеа за патеките

- **Споделување на рути** - корисниците можат да ги споделуваат креираните рути

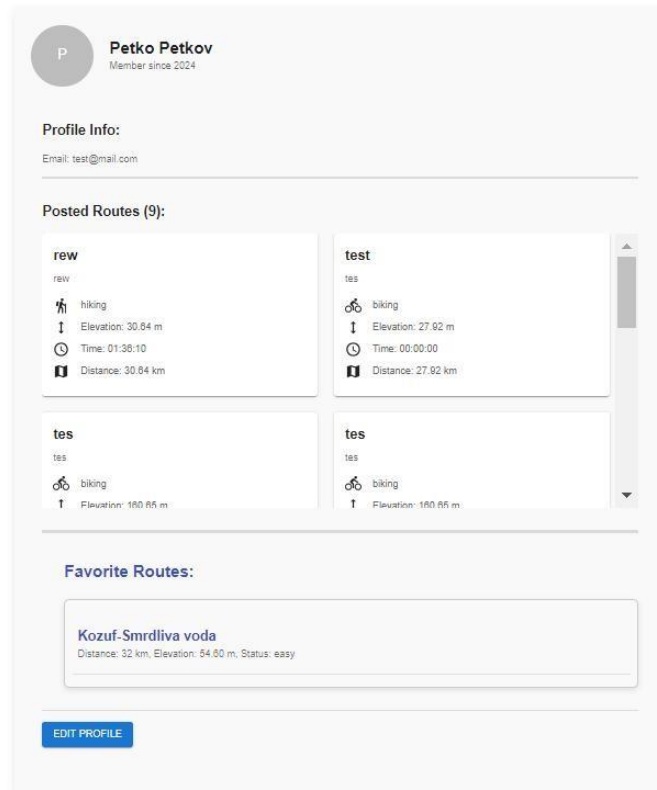
- **Преземање на рути** - корисниците можат да ги преземат рутите и да ги искористат на нивните GPS уреди



Слика 3. Детали за мапа, оставање коментари, споделување и преземање на рута

Исто така, овде може да се забележи мапа со елевација, креирана со помош на библиотеката **recharts**, која ги зема координатите од грх фајлот и ја исцртува елевацијата.

- **Кориснички профили** - корисниците можат да креираат свој профил на којшто може да ги видат основните кориснички информации, да ги прегледаат рутите кои тие ги објавиле и да ги видат зачуваните рути



Слика 4. Кориснички профил

### 3. Инсталација

За инсталација на проектот TrailVista потребен е **php 8.2** со **Laravel 11**. На следниве линкови може да се најдат инсталациите:

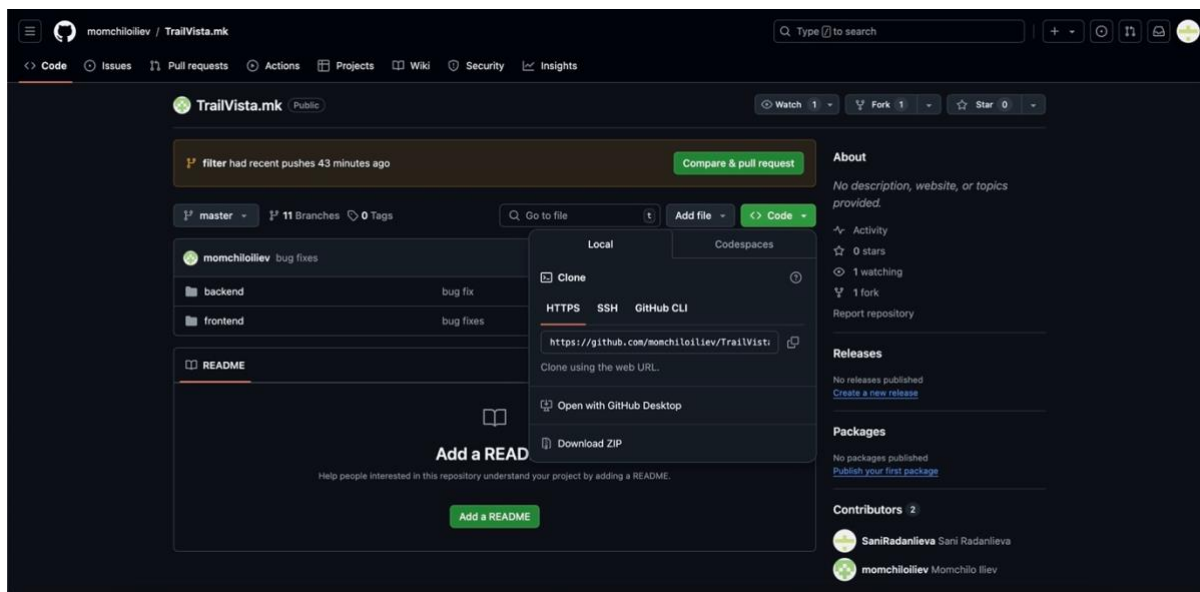
- За инсталација на php: <https://www.php.net/manual/en/install.php>
- За инсталација на composer: <https://getcomposer.org/doc/00-intro.md>
- За инсталација на npm: <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>
- За инсталација на PostgreSQL: <https://www.postgresql.org/download/>

### 4. Клонирање на репозиториумот

Клонирање на репозиториумот се врши преку еден од овие два линкови:

<https://github.com/momchiloiliev/TrailVista.mk>

<https://gitlab.finki.ukim.mk/ioss/trail-vista>



Слика 5. Клонирање на репозиториумот преку GitHub

За клонирање на проектот преку GitHub, потребно е да се избере клонирање преку HTTPS, SSH, GitHub CLI. На пример, преку користење на SSH, потребно е да се креира фолдер, да се копира SSH линкот, и локално во терминал во креирано фолдер потребно е да се изврши ***git clone -SSH <url>***.

За инсталација на зависности, потребно е да се влезе во директориумот каде што се наоѓа проектот. За инсталација на зависности потребни за backend, потребно е да се влезе во истоимениот директориум, и да се изврши командата ***composer install***. За инсталација на зависности потребни за frontend, потребно е да се влезе во истоимениот директориум, и да се изврши командата ***npm install***.

Следно, потребно е да се конфигурира ***.env*** фајлот, за користење на базата на податоци во рамки на проектот.

```
DB_CONNECTION=pgsql
DB_HOST=127.0.0.1
DB_PORT=5433
DB_DATABASE=
DB_USERNAME=
DB_PASSWORD=
```

Слика 5. Конфигурирање на ***.env***

По конфигурирање на базата на податоци и по успешно конектирање, потребно е да се креираат миграциите и шемата на базите на податоци, во backend фолдерот на проектот, преку командата ***php artisan migrate***.

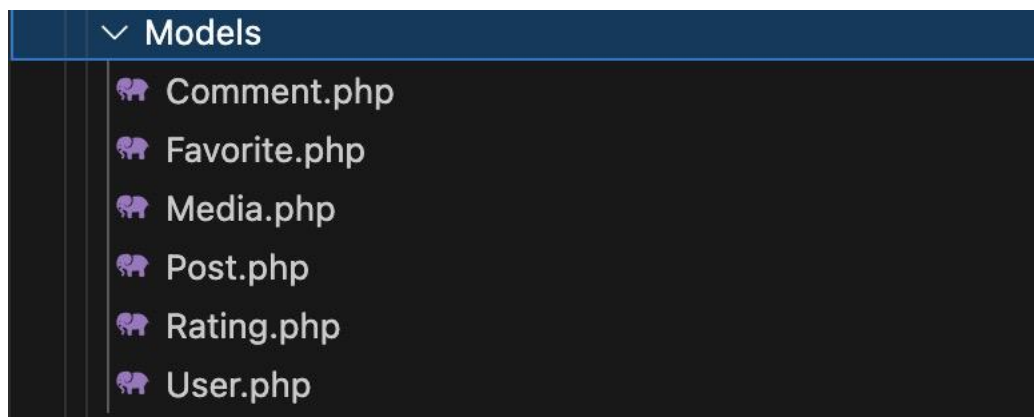
## 5. Архитектура на проектот

Овој проект има јасно дефинирана структура на директориуми и фајлови која ја олеснува организацијата на кодот и ресурсите. Подолу е преглед на основните директориуми и нивната намена:

- **app/** е главниот директориум на апликацијата, кој содржи модели, контролери, middleware и други класи.
- **bootstrap/** содржи фајлови за иницијализација на апликацијата. Фајлот `app.php` ја иницијализира Laravel рамката.
- **config/** содржи конфигурациски фајлови за апликацијата.
- **database/** вклучува миграции, фабрики и полначи за базата на податоци.
- **public/** е директориум од кој се сервираат јавни ресурси, како CSS, JavaScript и слики.
- **resources/** содржи ресурси како Blade темплејти, JavaScript фајлови, локализационски фајлови и assets.
- **routes/** е директориум кој ги содржи фајловите за дефинирање на апликациските рути, како `web.php`, `api.php`, и други.
- **storage/** се користи за складирање на компајлирани Blade темплејти, сесија, кеширање и фајлови генерирани од апликацијата.
- **tests/** е директориум за unit и feature тестови.
- **vendor/** содржи Composer пакети кои се инсталирани како зависности за проектот.

## 6. Модели

TrailVista.mk проектот содржи 6 модели, меѓу кои, Comment, Favorite, Media, Post, Rating, User.



Слика 6. Models

### 6.1. Модел: Comment

Моделот **Comment** претставува модел за коментари од корисниците на постовите на платформата TrailVista. Секој коментар е поврзан со одреден пост и корисник, и може да

содржи медиумски фајлови. Овој модел се користи за складирање и управување со коментарите од корисниците на различни патеки.

- **Структура на табелата**

- Име на табелата: comments
- Примарен клуч: id

- **Полиња кои се пополнуваат**

- Следните полиња можат да бидат масовно доделени преку `fillable` низата во моделот:

- post\_id: (цел број) ID на постот со кој е поврзан коментарот.
- user\_id: (цел број) ID на корисникот кој го оставил коментарот.
- author\_name: (стринг) Име на авторот на коментарот. Ова овозможува анонимни коментари или коментари од гости доколку е потребно.
- content: (текст) Содржината на коментарот од корисникот.

- **Врски (релации)**

- post(): Дефинира one-to-many релација кон моделот Post.
  - Секој коментар припаѓа на еден пост.
  - Методот belongsTo() на Laravel се користи за да покаже дека полето post\_id во табелата comments е надворешен клуч кој се однесува на полето id во табелата posts.
- user(): Дефинира one-to-many релација кон моделот User.
  - Секој коментар е направен од одреден корисник.
  - Методот belongsTo() на Laravel се користи за да покаже дека полето user\_id во табелата comments е надворешен клуч кој се однесува на полето id во табелата users.
- media(): Дефинира релација many-to-one кон моделот Media.
  - Секој коментар може да има повеќе медиумски фајлови поврзани со него, како слики или видеа кои ги прикачил корисникот.
  - Методот hasMany() се користи за да се дефинира дека коментарот може да има многу медиумски фајлови поврзани со него.

## 6.2. Модел: Favorite

Моделот **Favorite** претставува модел за означување на омилен постови од страна на корисниците на платформата TrailVista. Овој модел се користи за управување со листата на омилен постови на корисниците.

- **Структура на табелата**

- Име на табелата: favorites
- Примарен клуч: id

- **Полиња кои се пополнуваат**

- Следните полиња можат да бидат масовно доделени преку fillable низата во моделот:

- post\_id: (цел број) ID на постот кој е означен како омилен од корисникот.

- **Врски (релации)**

- post(): Дефинира one-to-many релација кон моделот Post.
  - Секој запис во Favorite табелата припаѓа на еден пост.



- Методот `belongsTo()` на Laravel се користи за да покаже дека полето `post_id` во табелата `favorites` е надворешен клуч кој се однесува на полето `id` во табелата `posts`.

### 6.3. Модел: Media

Моделот **Media** претставува модел за медиумски фајлови кои се прикачени на постовите во платформата TrailVista. Овој модел се користи за складирање и управување со слики, видео и други типови на медиумски фајлови поврзани со постовите.

- **Структура на табелата**

- Име на табелата: `media`
- Примарен клуч: `id`

- **Полиња кои се пополнуваат**

- Следните полиња можат да бидат масовно доделени преку `'fillable'` низата во моделот:

- `post_id`: (цел број) ID на постот со кој е поврзан медиумскиот фајл.
- `file_path`: (стринг) Локација на фајлот на серверот.
- `type`: (стринг) Тип на фајлот (на пример, слика, видео).

- **Врски (релации)**

- `post()`: Дефинира `one-to-many` релација кон моделот `Post`.

- Секој медиум припаѓа на еден пост.
- Методот `belongsTo()` на Laravel се користи за да покаже дека полето `post_id` во табелата `media` е надворешен клуч кој се однесува на полето `id` во табелата `posts`.

### 6.4. Модел: Post

Моделот **Post** претставува модел за постови на платформата TrailVista, кои ги опишуваат различните патеки што корисниците можат да ги додаваат. Секој пост е поврзан со одреден корисник и може да содржи коментари, оценки, медиумски фајлови и може да биде означен како омилен од корисниците.

- **Структура на табелата**

- Име на табелата: `posts`
- Примарен клуч: `id`

- **Полиња кои се пополнуваат**

- Следните полиња можат да бидат масовно доделени преку `'fillable'` низата во моделот:

- `user_id`: (цел број) ID на корисникот кој го создал постот.
- `author_name`: (стринг) Име на авторот на постот.
- `title`: (стринг) Наслов на постот.
- `description`: (текст) Опис на постот.
- `moderation_status`: (стринг) Статус на модерација на постот (дали е одобрен или не).



- sport: (стринг) Спортската категорија на патеката (на пример: планинарење, велосипедизам, трчање).
- file\_path: (стринг) Локација на GPX фајлот или друг придружен фајл на серверот.
- elevation: (цел број) Вкупно издигнување на патеката (во метри).
- distance: (цел број) Вкупна должина на патеката (во километри).
- time: (цел број) Времетраење на патеката (во минути).

#### • Врски (релации)

- user(): Дефинира one-to-many релација кон моделот User.
  - Секој пост припаѓа на еден корисник.
  - Методот belongsTo() се користи за да покаже дека полето user\_id е надворешен клуч кој се однесува на полето id во табелата users.
- comments(): Дефинира many-to-one релација кон моделот Comment.
  - Секој пост може да има повеќе коментари.
  - Методот hasMany() се користи за да се дефинира дека постот може да има многу поврзани коментари.
- ratings(): Дефинира many-to-one релација кон моделот Rating.
  - Секој пост може да има повеќе оценки (рејтинзи) од корисниците.
  - Методот hasMany() се користи за да се дефинира дека постот може да има многу поврзани оценки.
- media(): Дефинира many-to-one релација кон моделот Media.
  - Секој пост може да има повеќе медиумски фајлови поврзани со него, како слики или видеа.
  - Методот hasMany() се користи за да се дефинира дека постот може да има многу поврзани медиумски фајлови.
- favorites(): Дефинира many-to-one релација кон моделот Favorite.
  - Секој пост може да биде означен како омилен од повеќе корисници.
  - Методот hasMany() се користи за да се дефинира дека постот може да има многу поврзани омилени записи.

## 6.5. Модел: User

Моделот **User** претставува корисник на платформата TrailVista. Овој модел се користи за автентикација на корисниците, складирање на нивните податоци и управување со нивните постови, омилени патеки и други активности. Корисниците исто така можат да имаат администраторски права.

#### • Структура на табелата

- Име на табелата: users
- Примарен клуч: id

#### • Полиња кои се пополнуваат

- Следните полиња можат да бидат масовно доделени преку `fillable` низата во моделот:

- name: (стринг) Името на корисникот.
- email: (стринг) Е-мејл адреса на корисникот.
- password: (стринг) Хеширана лозинка на корисникот.

- **Скриени полиња**

- Овие полиња нема да бидат видливи кога податоците од моделот се претставуваат во JSON формат:

- password: (string) Хеширана лозинка.
    - remember\_token: (string) Токен за запомнување на корисникот при следни пријавувања.

- **Полиња кои се кастираат**

- Следните полиња се автоматски претворени во специфични типови на податоци:
    - email\_verified\_at: (datetime) Датум и време на верификација на е-мејл адресата.
    - password: (hashed) Лозинката е автоматски хеширана.

- **Врски (релации)**

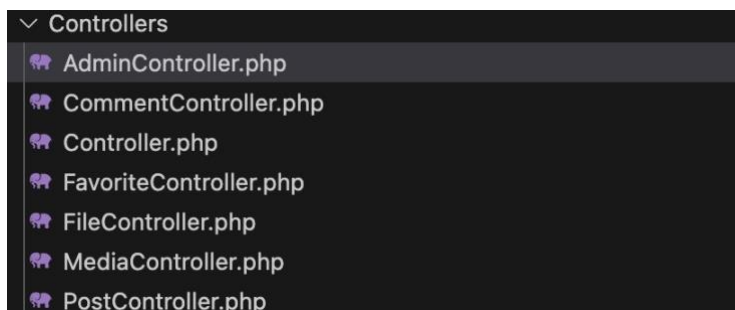
- posts(): Дефинира one-to-many релација кон моделот Post.
    - Секој корисник може да има повеќе постови.
    - Методот hasMany() се користи за да покаже дека корисникот може да има многу поврзани постови.
  - favorites(): Дефинира many-to-many релација кон моделот Post преку табелата `favorites`.
    - Корисникот може да има повеќе омилен постови.
    - Методот belongsToMany() се користи за да се дефинира дека корисникот може да ги означат повеќе постови како омилен преку табелата `favorites`.
    - Оваа релација користи timestamp за да бележи кога постот е означен како омилен.

- **Дополнителни функции**

- isAdmin(): Проверува дали корисникот има администраторски привилегии.
    - Враќа true или false во зависност од вредноста на полето `isAdmin` (прилагодете го според вашата база на податоци).
    - Оваа функција овозможува разликување на администратори од обични корисници.

## 7. Контролери

Имаме 7 контролери, како CommentController, Controller, FavoriteController, FileController, MediaController, PostController, AdminController, .



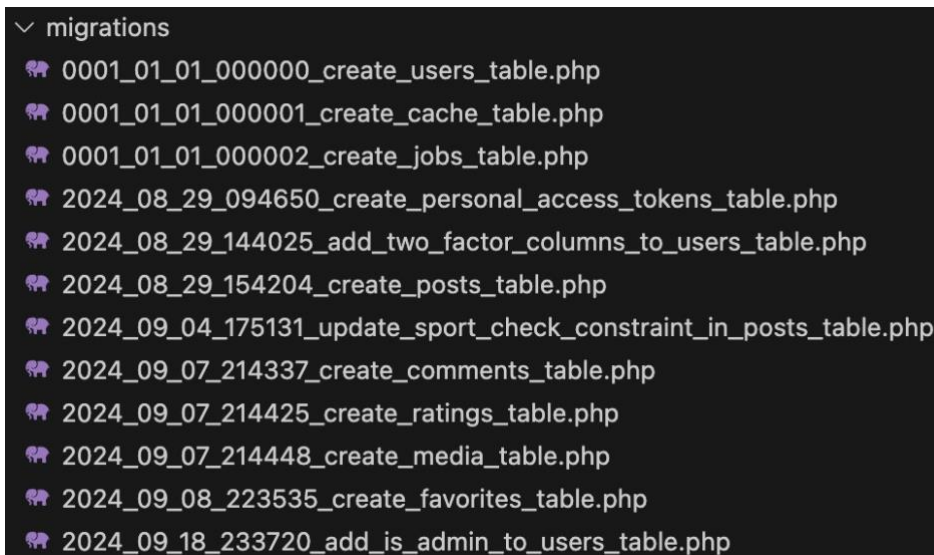
Слика 7. Controllers

- Контролерот **`AdminController`** управува со администраторските функции во платформата. Тој овозможува листање на сите постови и корисници, како и бришење на специфичен пост или корисник по даден ID. Сите операции враќаат JSON одговор со статус на успех и релевантна порака.

- Контролерот **CommentController** е одговорен за управување со коментарите на постовите. Методот `index` ги прикажува сите коментари поврзани со одреден пост преку неговиот `post_id` и ги враќа како JSON одговор. Методот `store` овозможува креирање на нов коментар поврзан со постот, со валидирање на содржината на коментарот, и го враќа новокреираниот коментар како JSON одговор со статус 201 (креирано).
- Контролерот **Controller** е базен контролер кој го користат сите други контролери во апликацијата. Тој ги имплементира функционалностите за авторизација преку `AuthorizesRequests` и за валидација преку `ValidatesRequests`. Овој контролер служи како основа за наследување од сите други контролери, обезбедувајќи им унифициран начин за авторизација и валидација на барања.
- Контролерот **FavoriteController** е одговорен за управување со омилените постови на корисниците. Тој овозможува додавање и отстранување на постови од листата на омилените постови на тековниот најавен корисник, прикажување на сите омилените постови на корисникот, како и приказ на најфаворизираниите патеки.
  - `addToFavorites($postId)`: Додава одреден пост во омилените на корисникот, доколку тој не е веќе додаден.
  - `removeFromFavorites($postId)`: Отстранува одреден пост од омилените на корисникот, ако тој веќе е во листата.
  - `getFavorites()`: Ги враќа сите омилените постови на тековниот корисник, заедно со информациите за авторите на тие постови.
  - `mostFavoritedRoutes()`: Ги прикажува петте најфаворизирани патеки на платформата, сортирани според бројот на омилените записи.
- Контролерот **FileController** е одговорен за преземање на GPX фајлови од складиштето. Методот `getGpxFile` проверува дали фајлот постои, а потоа го презема со соодветно генерирано име и поставен Content-Type.
- Контролерот **MediaController** управува со медиумските фајлови поврзани со постовите. Методот `store` овозможува прикачување и складирање на медиумски фајлови (слики и видеа) за одреден пост. Методот `index` ги враќа сите медиумски фајлови поврзани со одреден пост, а доколку нема пронајдени фајлови, враќа порака со статус 404.
- Контролерот **PostController** управува со постовите во апликацијата. Методот `index` ги враќа сите постови заедно со корисникот кој ги креирал. Методот `show` прикажува детали за специфичен пост, вклучувајќи коментари, оценки и медиумски фајлови. Методот `store` креира нов пост со валидирање на податоците и прикачување на GPX фајл. Методот `update` овозможува уредување на постови, а методот `destroy` овозможува бришење на постови, при што се проверува дали корисникот е автор на постот.

## 8. Миграции и рути

Миграциите кои беа креирани се следниве:



```
▼ migrations
🐘 0001_01_01_000000_create_users_table.php
🐘 0001_01_01_000001_create_cache_table.php
🐘 0001_01_01_000002_create_jobs_table.php
🐘 2024_08_29_094650_create_personal_access_tokens_table.php
🐘 2024_08_29_144025_add_two_factor_columns_to_users_table.php
🐘 2024_08_29_154204_create_posts_table.php
🐘 2024_09_04_175131_update_sport_check_constraint_in_posts_table.php
🐘 2024_09_07_214337_create_comments_table.php
🐘 2024_09_07_214425_create_ratings_table.php
🐘 2024_09_07_214448_create_media_table.php
🐘 2024_09_08_223535_create_favorites_table.php
🐘 2024_09_18_233720_add_is_admin_to_users_table.php
```

Слика 8. Migrations

Миграциите во проектот се користат за креирање и ажурирање на базата на податоци.

- create\_users\_table: Креира табела за корисници.
- create\_cache\_table: Креира табела за кеширање.
- create\_jobs\_table: Креира табела за позадински задачи.
- create\_personal\_access\_tokens\_table: Креира табела за лични пристапни токени.
- add\_two\_factor\_columns\_to\_users\_table: Додава колони за двофакторска автентикација на табелата за корисници.

- create\_posts\_table: Креира табела за постови.  
- update\_sport\_check\_constraint\_in\_posts\_table: Ажурирање на ограничување за спорт во табелата за постови.

- create\_comments\_table: Креира табела за коментари.
- create\_ratings\_table: Креира табела за оценки.
- create\_media\_table: Креира табела за медиумски фајлови.
- create\_favorites\_table: Креира табела за омилените постови.
- add\_is\_admin\_to\_users\_table: Додава поле за администраторски права во табелата за корисници.

На следните слики може да се разгледаат рутите кои се користат во рамки на проектот:



```
//POSTS
Route::get('/posts', [PostController::class, 'index']);
Route::get('/posts/{id}', [PostController::class, 'show']);
// Route::post('/posts', [PostController::class, 'store']);

//POSTS COMMENTS RATINGS MEDIA
Route::get('/posts/{postId}/comments', [CommentController::class, 'index']);
Route::get('/posts/{postId}/ratings', [RatingController::class, 'index']);
Route::get('/posts/{postId}/media', [MediaController::class, 'index']);

//MOST FAVORITED ROUTES
Route::get('/most-favorited-routes', [FavoriteController::class, 'mostFavoritedRoutes']);

//GPX FILES
Route::get('/storage/gpx-files/{filename}', [FileController::class, 'getGpxFile']);
Route::get('/download-gpx/{filename}/{title}', [FileController::class, 'getGpxFile']);
```

Слика 9. Јавни рути

```
// Protected routes (Require authentication)
Route::middleware('auth:sanctum')
->group(function () {

    //GET USER DATA
    Route::get('/user', function (Request $request) {
        return new UserResource($request->user());
    });

    //POSTS
    Route::post('/posts', [PostController::class, 'store']); // Add a new post
    Route::put('/posts/{id}', [PostController::class, 'update']); // Edit a post
    Route::delete('/posts/{id}', [PostController::class, 'destroy']); // Delete a post

    //POSTS COMMENTS RATINGS MEDIA
    Route::post('/posts/{postId}/comments', [CommentController::class, 'store']);
    Route::post('/posts/{postId}/ratings', [RatingController::class, 'store']);
    Route::post('/posts/{postId}/media', [MediaController::class, 'store']);

    //FAVORITES
    Route::post('/favorites/{postId}', [FavoriteController::class, 'addToFavorites']);
    Route::delete('/favorites/{postId}', [FavoriteController::class, 'removeFromFavorites']);
    Route::get('/favorites', [FavoriteController::class, 'getFavorites']);

});
```

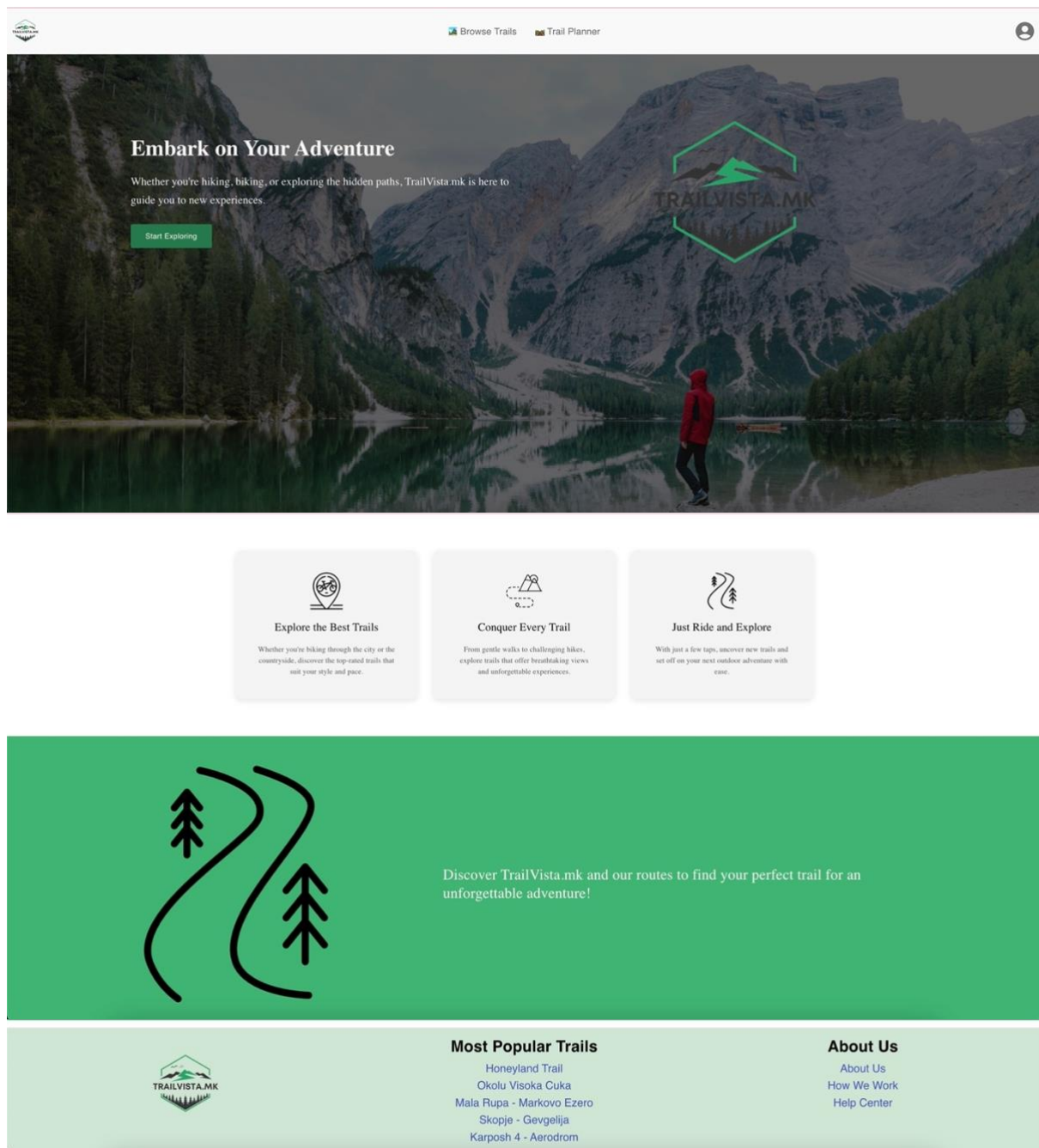
Слика 10. Руте кои бараат автентикација

Овој фајл дефинира рутите за API на апликацијата:

- јавни руте (без автентикација):
  - /status: проверка на статусот на API.
  - /posts: прикажување на сите постови.
  - /posts/{id}: прикажување на детали за одреден пост.
  - /posts/{postId}/comments: прикажување на коментари за одреден пост.
  - /posts/{postId}/ratings: прикажување на оценки за одреден пост.
  - /posts/{postId}/media: прикажување на медиумски фајлови за одреден пост.
  - /most-favorited-routes: прикажување на најфаворизираните патеки.
  - /storage/gpx-files/{filename}: преземање на GPX фајл.
  - /download-gpx/{filename}/{title}: преземање на GPX фајл со специфично име.
- заштитени руте (бараат автентикација преку sanctum):
  - /user: прикажување на податоци за најавениот корисник.
  - /posts: додавање на нов пост.
  - /posts/{id}: ажурирање на пост.
  - /posts/{id}: бришење на пост.
  - /posts/{postId}/comments: додавање на коментар на пост.
  - /posts/{postId}/ratings: додавање на оценка на пост.
  - /posts/{postId}/media: додавање на медиумски фајлови на пост.
  - /favorites/{postId}: додавање на пост во омилени.
  - /favorites/{postId}: отстранување на пост од омилени.
  - /favorites: прикажување на омилени постови на корисникот.

## 9. Преглед на апликацијата

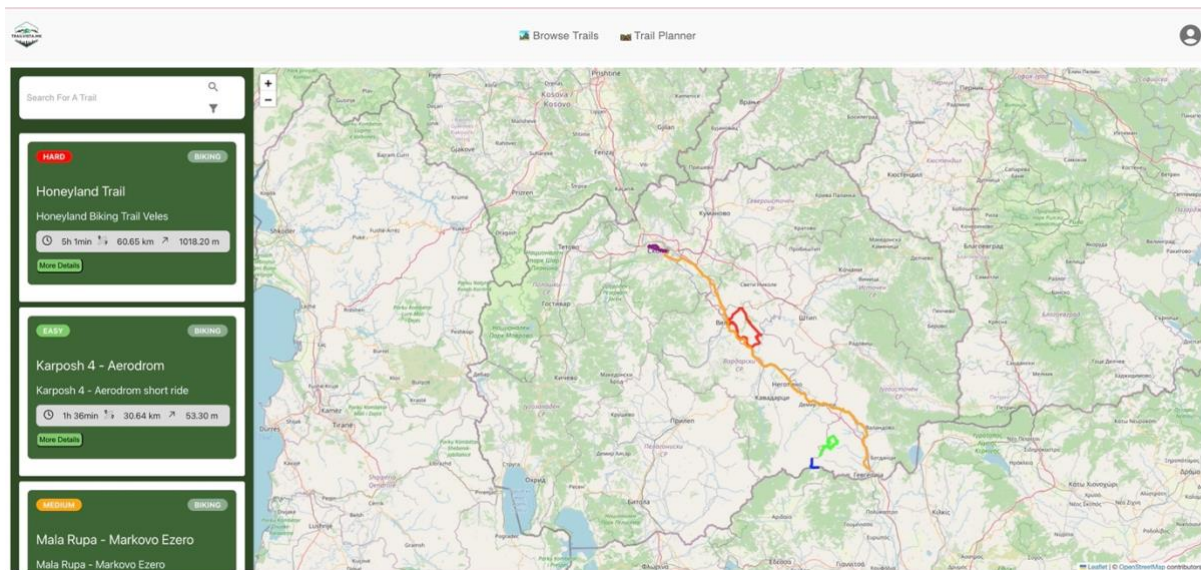
Веб апликацијата TrailVista.mk има почетна страница, на која може да се види header со мени за барање на рути, додавање на нова рута, како и login/register дел (или мени за менаџирање на профил доколку корисникот е автентизиран).



Слика 11. Преглед на landing page

Во footer делот на оваа страница може да се видат најпопуларните рути. Тоа се рутите кои најмногу корисници ги имаат забележано како омилен.





Слика 12. Преглед на Browse Trails

На Browse Trails може да се видат лево сите рути кои се додадени на платформата, со нивно име, опис, тежина на рута, вид на спорт за кој е наменета рутата, време на движење, должина во километри, и надморска висина - елевација прикажана во метри. Исто така, може да се видат повеќе детали за секоја рута посебно.

На оваа страна имаме мапа на која се исцртани сите додадени мапи, секоја во различна боја.

Најпрво се преземаат податоците за сите рути со `useEffect()`, преку повик до API-то (`/api/posts`), а откако ќе се добијат податоците за рутите, за секоја руа се прави дополнителен повик до серверот за преземање на GPX фајлот.

Преземениот GPX фајл се парсира со помош на `DOMParser`, кој го претвара GPX XML во документ, од каде се извлекуваат координатите. Секој GPX фајл содржи елементи со GPS податоци како `track points` - секој од овие `track points` (`trkpt`) содржи атрибути широчина и должина кои ја претставуваат точната географска локација на патеката во тој момент.

Имаме функција `parseGPX` која е задолжена за сето ова и истата го итерира секој `track point` елемент во низата `trkpts`. земајќи ги атрибутите `lat` и `lon`, кои се стрингови кои ги конвертираме за да можеме да ги употребиме како координати.

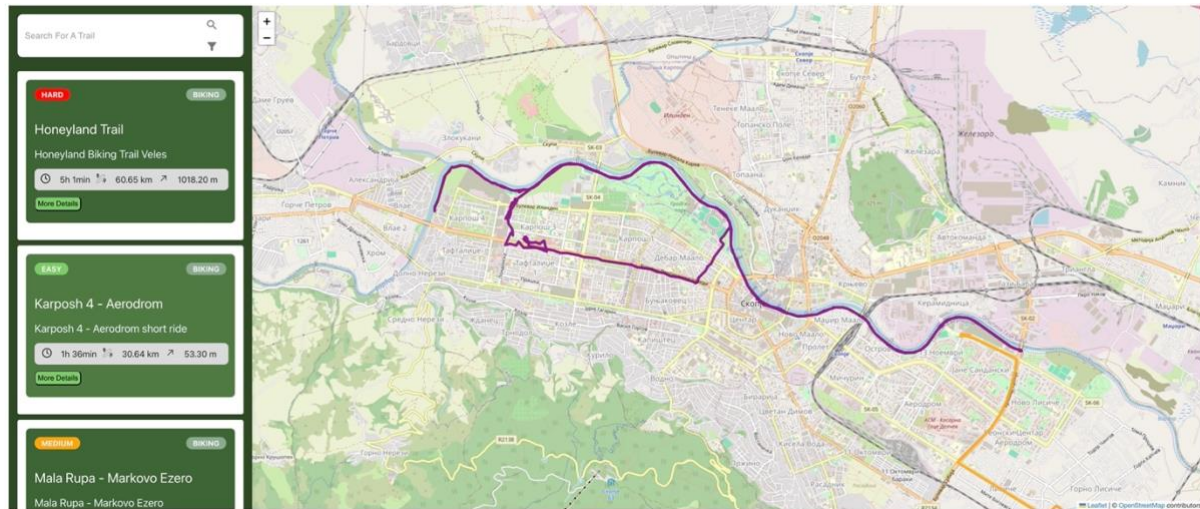
Извлечените координати се складираат во низа како двојки, а секоја точка се зачувува како `LatLngTuple` - тоа е низата која мапата ја користи за исцртување на линијата со помош на `Polyline`.

```
// Parse GPX file and extract coordinates
const parseGPX = (gpxData: Document) => {
  const trkpts = gpxData.getElementsByTagName('trkpt');
  const coordinates: LatLngTuple[] = [];
  for (let i = 0; i < trkpts.length; i++) {
    const lat = parseFloat(trkpts[i].getAttribute('lat') || '0');
    const lon = parseFloat(trkpts[i].getAttribute('lon') || '0');
    coordinates.push([lat, lon]);
  }
  return coordinates;
};
```

Слика 13. Функција `parseGPX`

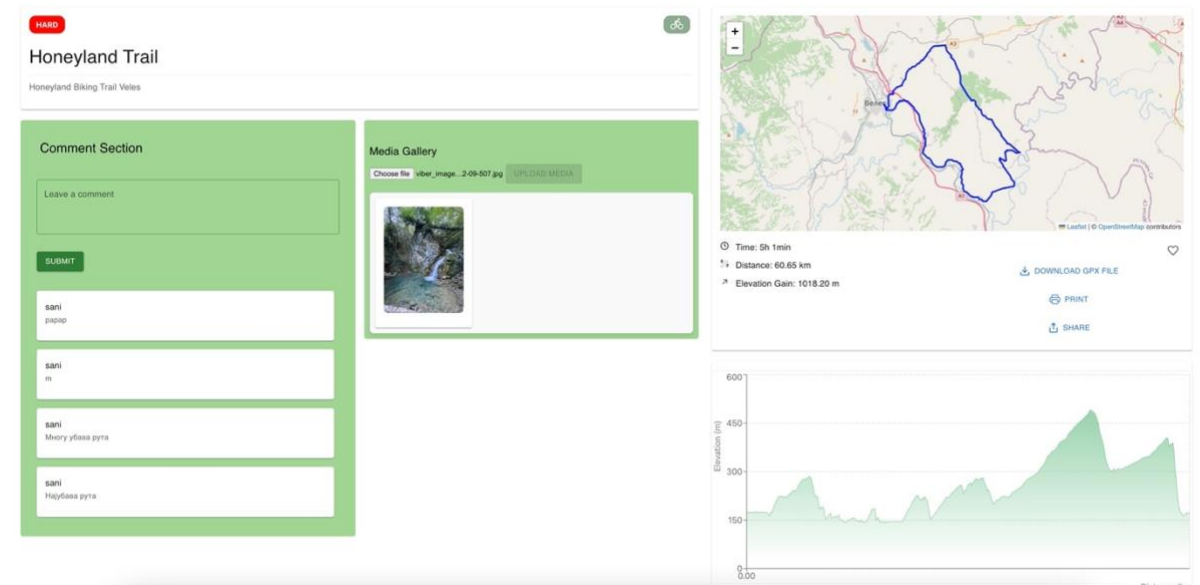


Корисникот може да избере некоја од додадените рути и при клик врз картичката со таа рута, истата се прикажува сама на мапата. Ова е овозможено со рецентрирање на мапата според координатите на GPX фајлот.



Слика 14. Приказ на одредена рута

Доколку корисникот одбере да разгледа повеќе детали за одредена рута, при клик на копчето See Details, го носи корисникот на посебна страна во која се прикажани повеќе детали за рутата.



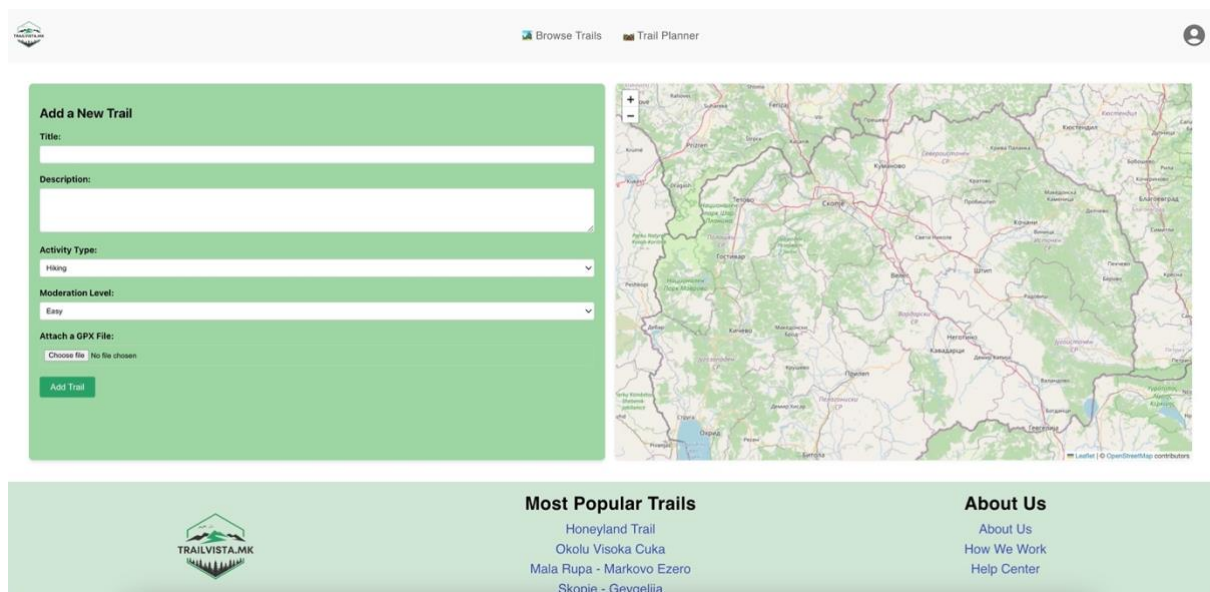
Слика 15. Приказ на детали за одредена рута

Овде може исто така да се видат основните информации за избраната рута. Исто така, имаме секција со коментари и секција со галерија, при што само автентифицирани корисници можат да оставаат коментари и да поставуваат слики во галеријата. Десно повторно е прикажана рутата на мапата. Овде имаме опција да ја додадеме оваа рута во омилени, да го симнеме локално грх фајлот, да ја испринтаме целата страница или да копираме линк за споделување на оваа рута.

Исто така, овде се наоѓа и мапата со елевација, која ја креираме базирано на GPX податоците и библиотеката **recharts**. Најпрво се парсира GPX фајлот со DOMParser за да се извлечат координатите и елевацијата. Преку функцијата calculateDistance се пресметува растојанието меѓу тековната и претходната точка, и тоа растојание се сумира за да се добие вкупното растојание од почетната до тековната точка. Функцијата calculateDistance пресметува растојание меѓу две GPS координати со помош на формулата на Хаверсајн, која овозможува точно пресметување на растојание врз основа на географската широчина и должина.

Податоците за секоја точка, како растојание и елевација се чуваат во низа наречена elevationData, која потоа се користи за креирање на графиконот.

Од библиотеката recharts се користи AreaChart, се дефинираат оски и се прикажува елевацијата како Area графикон со прилагоден градиент. Се користи и tooltip за да се прикаже информација за точната елевација и растојание кога корисникот ќе го задржи глумчето на одредена точка од графиконот.



Слика 16. Додавање на нова рута

Корисниците кои се автентифицирани може да додадат нова рута со GPX фајл, име, опис, тип на активност, тежина на рута. Кога корисникот ќе додаде рута со GPX фајл, истата се исцртува на мапата десно.



Слика 17. Приказ на рута на мапа при додавање

Корисниците можат да управуваат со своите кориснички профили, да видат повеќе информации, како што е меил на корисник, име на корисник, од кога е зачленет на платформата и слично.

Овде исто така може да се видат рутите кои тој корисник ги има додадено, а може да се видат и омилените рути на корисникот.

P

**sani**  
Member since 2024

**Profile Info:**  
Email: lola@gmail.com

**Posted Routes (5):**

**Honeyland Trail**  
Honeyland Biking Trail Veles  
🚴 biking  
⬆ Elevation: 60.65 m  
🕒 Time: 05:01:49  
📖 Distance: 60.65 km

**Okolu Visoka Cuka**  
Asan cesma - Volcka cesma - Konjare - Asan cesma  
🚴 biking  
⬆ Elevation: 44.43 m  
🕒 Time: 03:59:39  
📖 Distance: 44.43 km

**Mala Rupa - Markovo Ezero**  
Mala Rupa - Markovo Ezero  
🚴 biking  
⬆ Elevation: 18.41 m

**Skopje - Gevgelija**  
Skopje - Gevgelija Cycling Trail  
🚴 biking  
⬆ Elevation: 160.65 m

**Favorite Routes:**

**Honeyland Trail**  
Distance: 61 km, Elevation: 1018.20 m, Status: hard

EDIT PROFILE

Слика 18. Приказ на кориснички профил

Исто така, на платформата има админ дел. Доколку некој корисник е администратор, може да пристапи на /admin патека и да ги прегледа сите постови, и корисници. Администраторот може да избрише пост, да разгледа пост и да избрише постоечки корисник.