Web Technologies
WorldSkills 2024 National Competition HUNGARY

Round 2

Submitted by:
Skills IT

WS2024_HU_TP17_R2_EN

Contents

Introduction

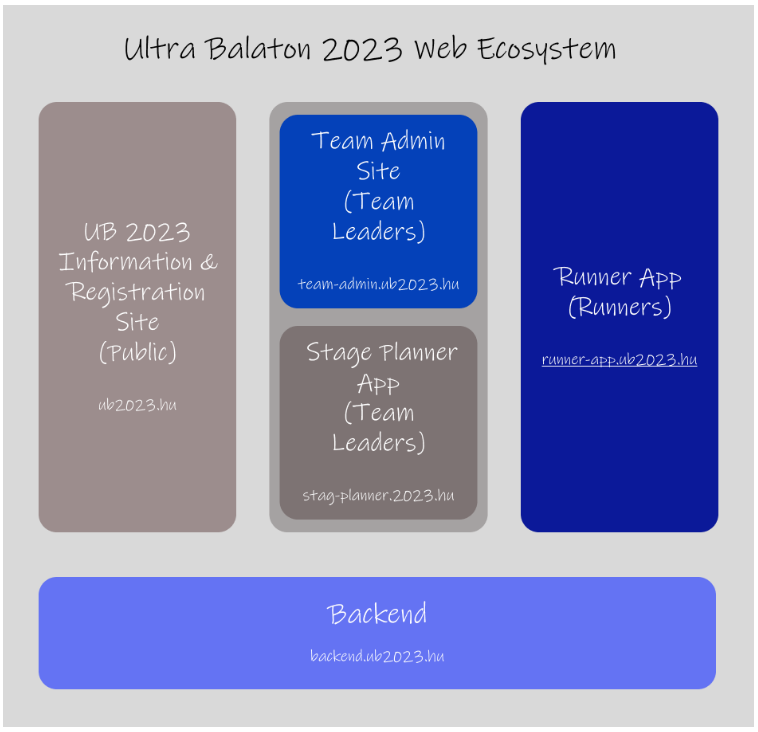
You used to work as a freelance web developer, but now you've applied for a job as a developer at a large software development company. The company's management wants to test your skills, so as part of the recruitment process, they asked you to develop a prototype web application for the Ultra Balaton (UB) Running Festival.

In the first round of the recruitment process, you were asked to develop a prototype of the information page for the festival and the stage calculator for the team category. You have successfully completed this round, as you have been selected from more than 100 applicants to be one of the 6 people who will be tested by the management with a new task.

Description of project and tasks

In this new round, you will need to create three element of the Ultra Balaton 2023 web ecosystem:

- a REST API backend module to serve and manage the data stored in the database (Backend, Part 1)
- an administration module for team leaders to manage their own team (Team Admin Site, Part 2)
- a mobile-optimised web application for runners to track the team's running and record the time of handovers (Runner App, Part 3).


Ultra Balaton 2023 Web Ecosystem
UB 2023 Information & Registration Site (Public)
ub2023.hu
Team Admin Site (Team Leaders)
team-admin.ub2023.hu
Stage Planner App (Team Leaders)
stag-planner.2023.hu
Runner App (Runners)
runner-app.ub2023.hu
Backend
backend.ub2023.hu

How to submit your work

1. **Part 1 – Backend**
    a. If you are creating the backend in native PHP, place `index.php` and other PHP source files in the `d:\ws2024-s17-hu-r2\submitted\deploy\backend.ub2023.hu\public` folder.
    b. If you are creating the backend in Laravel, move the contents of the `d:\ws2024-s17-hu-r2\assets\laravel` folder to the `d:\ws2024-s17-hu-r2\submitted\deploy\backend.ub2023.hu` folder and edit the code there.
    c. If you are creating the backend in Node.js/Express, move the contents of the `d:\ws2024-s17-hu-r2\assets\nodejs-app` folder to the `d:\ws2024-s17-hu-r2\submitted\deploy\backend.ub2023.hu` folder and edit the code there (the Node.js/Express backend must also be serving on port 80, so you will need to shut down the Apache server in this case.)
2. **Part 2 – Team Admin Site**
    a. If you are implementing the admin module with Vanila JS, create and edit your HTML\JS\CSS code in the the `d:\ws2024-s17-hu-r2\submitted\deploy\team-admin.ub2023.hu`
    b. If the admin module is created in one of the JS frameworks (React, Vue or Angular), move the contents of the corresponding project folder from `d:\ws2024-s17-hu-r2\assets` to `d:\ws2024-s17-hu-r2\submitted\dev\team-admin.ub2023.hu` and edit the code there. Copy your built project to `d:\ws2024-s17-hu-r2\submitted\deploy\team-admin.ub2023.hu` folder
3. **Part 3 – Runner App**
    a. If you are implementing the app with Vanila JS, create and edit your HTML\JS\CSS code in the the `d:\ws2024-s17-hu-r2\submitted\deploy\runner-app.ub2023.hu`
    b. If the app is created in one of the JS frameworks (React, Vue or Angular), move the contents of the corresponding project folder from `d:\ws2024-s17-hu-r2\assets` to `d:\ws2024-s17-hu-r2\submitted\dev\runner-app.ub2023.hu` and edit the code there. Copy your built project to `d:\ws2024-s17-hu-r2\submitted\deploy\runner-app.ub2023.hu` folder
4. At the end of the corresponding phases your work should be reachable on the following URLs:
    a. Backend REST API endpoint base URL: http://backend.ub2023.hu/api/v1
    b. Admin: http://team-admin.ub2023.hu
    c. Frontend: http://runner-app.ub2023.hu
5. Document the technology you have chosen for each task in the README.md file.
6. If you have not managed to deploy as described above, please describe in README.md exactly how we can check your work in each part. E.g.:

```
3. Runner App:
 - React
 - Source: d:\ws2024-s17-hu-r2\submitted\dev\runner-app.ub2023.hu
 - URL: http://localhost:3000
 - Additional instruction: issue npm run start command from the folder
above
```

Part 1 – Backend

In this part, you need to create a REST API server module with a few endpoints. The endpoints will be used by the different frontend modules (team admin, runner app etc.)

Authentication

All endpoints, except the /stages and the /login endpoint need to have authentication. The authentication should be done using bearer tokens in the “authorization” header. The bearer token contains the 9-digit numeric token of the user (it is generated when a new user is created). You need to send it with every request, using the following format: “Authorization: Bearer <token of the user>”

Description

For a full description of the endpoints to be created, see Appendix A.

The backend will serve using the following base URL: http://backend.ub2023.hu/api/v1

The database is already available with the data already uploaded:

- database name: ub2023
- username: ub2023
- password: admin

Part 2 – Team Admin Site

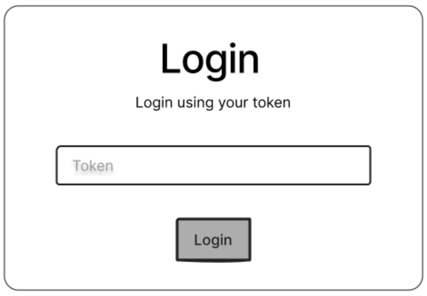
In this part, you need to create an administration module for team leaders to manage their own team.

We provide a full working backend solution for you before this task. You can reach the endpoints of this backend with the following base URL: http://backend2.ub2023.hu/api/v1. See Appendix A for details of the available endpoints.

The admin interface in this prototype version does not need to be responsive, the evaluation is done on a full HD desktop (1920x1080) display. However, in desktop view, it should follow the structures of the wireframes provided.

Team leader login

To use the admin module, the team leaders have to authenticate themselves by a 9-digit numeric token. If the user has not yet entered their token or the stored token cannot be identified by the backend (401 Unauthorized response from the backend), a pop-up window will appear or the user gets redirected to a login page allowing the user to enter their token.



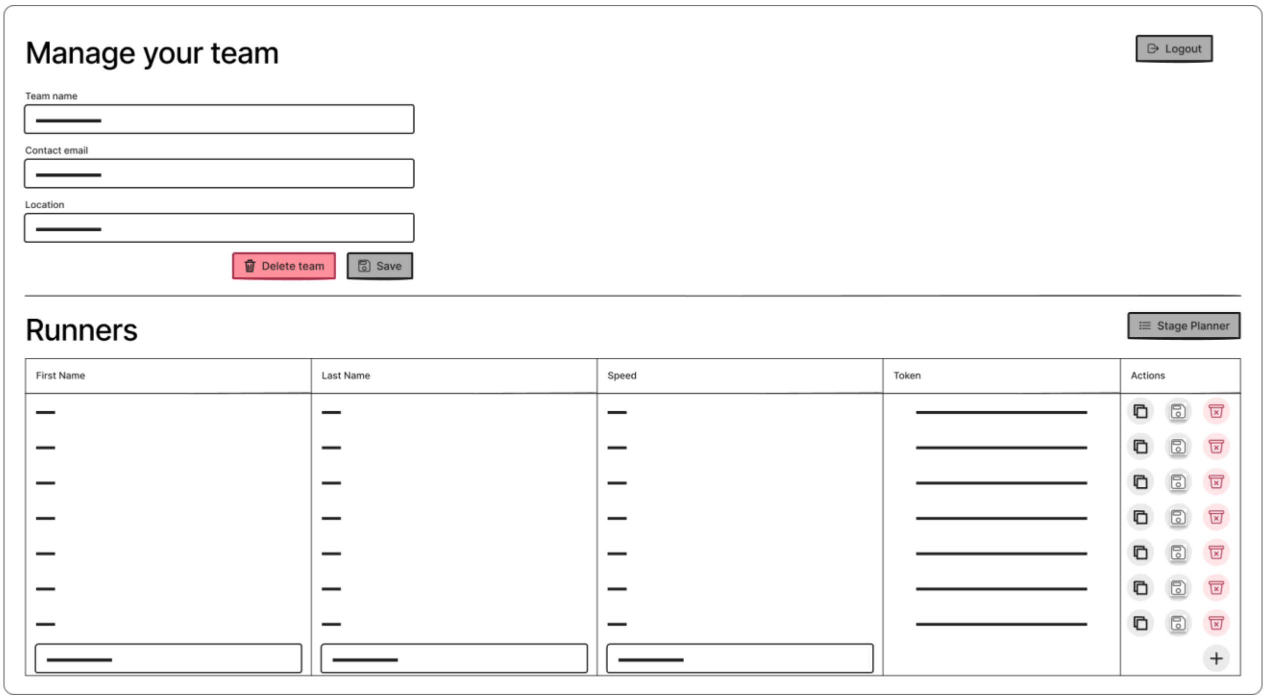
The token approved by the backend must be stored. It should be available when the browser is refreshed or restarted after closing so the user does not have to re-enter it.

Currently only one team leader account is stored in the database with the following token: 111111111

All requests to the backend must include the token in the Authorization header as a Bearer token. *(“Authorization: Bearer <token>”)*

After login the admin user will be redirected to the Team page.

Team page



On the Team page the team leader can edit the details of their team.

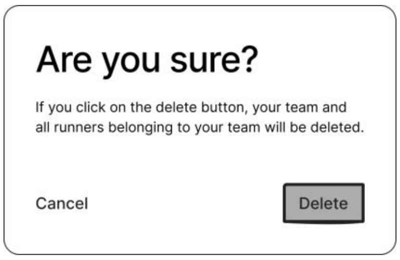
A Logout button appears on the top right corner of the window. Clicking this will log the user out and redirect them back to the login window.

In the top part of the page, you can see the basic team data in an editable form with the following fields:

- Team name
- Contact e-mail
- Location

Below the fields there is a “Save” button, which is only active if a field value has changed. Clicking the “Save” button saves the changes in the database.

A “Delete team” button appears next to the Save button. If you click on the “Delete team” button, a confirmation window appears asking "Are you sure? If you click on the delete button, your team and all runners belonging to your team will be deleted.”. (You can use the /resetDb endpoint, to recover the initial state, after deleting a team)


Are you sure?
If you click on the delete button, your team and all runners belonging to your team will be deleted.
Cancel
Delete

In the second half of the page, you can see the details of runners belonging to this team with fields that can be edited inline.

The runners have the following fields:

- First Name
- Last Name
- Speed (MM:SS, the estimated time it takes the runner to run 1 km)
- Token (9-digit number)

If a runner's details change, clicking on the Save icon at the end of the row will save the change. Click on the Trash icon at the end of the row to remove the team member from the team. If you click on the Copy icon, the runner's token is copied to the clipboard.

The token is generated by the backend, so this field is not editable.

After the last team member there is a row of empty fields. If you fill in the First Name and Last Name field of a new team member and then click the + button at the end of the row, the new team member's details will be saved, the generated token will get displayed and a new blank row will appear below.

A maximum of 10 team members can be added in this way.

There is also an additional "Stage Planner" button in the top left corner of the Runners section. This button will open the Stage Planner in a new browser window at http://stage-planner.ub2023.hu URL. (You may already be familiar with the Stage Planner from the first round. You do not need to create it this time, the planner is already available at the URL above.)

Note: To make testing easier, our backend randomly creates a new stage scheduler every time there is a change in the runner data (adding a new runner, deleting a runner, changing a runner's data).

Part 3 – Frontend

In this section, you will create a mobile-optimized web application that runners will use to easily track events during the competition and to record handover times for their own runs.

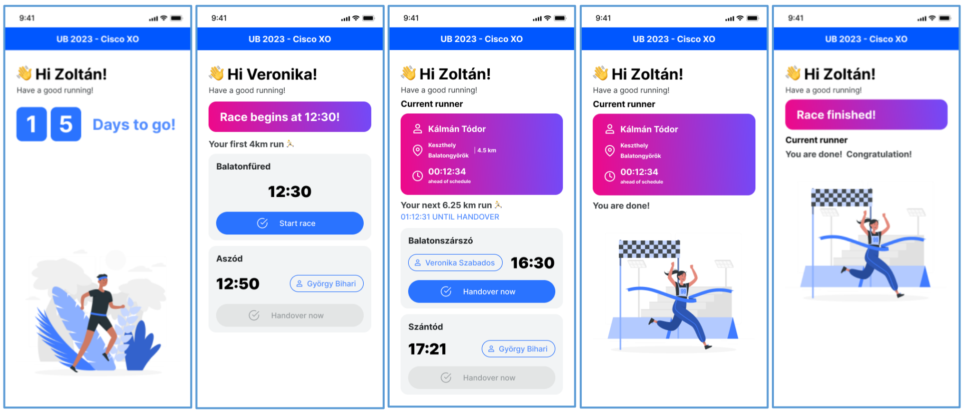
To perform this task, you have the backend already used in the previous task (https://backend2.ub2023.hu/api/v1).

Description

The application page has 5 main states:

1. In the days before the race
2. On the day of the race, before the start of the team
3. On the day of the race, after the start of the team

4. After the current runner’s last stage
5. After the team's run is complete



Testing

To make testing easier, you need to provide a way for setting a “fake date”. This means that a URL parameter called “fakeDate” should be added, which can receive an ISO date string. If this is provided, the application should use this date, instead of the current date for any date calculations.

For example, the URL should look like this, when using fake date: “http://runner-app.ub2023.hu?fakeDate=2023-04-01T09:00:00Z”.

General expectations on appearance

- Since runners will be using the app on their mobile devices, you need to create a mobile-optimized version of the app. We will not test your work in desktop view. The primary testing environment is iPhone 14.
- During development, you should follow the design given in the mockups as much as you can. As a help, you can find the style guide of the application in Appendix B.
- The name of the team is always displayed at the top of the window. Below this, a welcome text with the first name of the runner appears:
  Hi Zoltan!
  Have a good running!

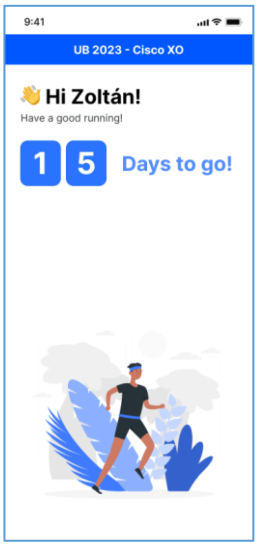
Sign in to the app

Logging in to the app is simply done by using the runner's own personalized link, which is in the following format:

http://runner-app.ub2023.hu?token=123456789

The token approved by the backend must be stored. It should be available when the browser is refreshed or restarted after closing so the user does not have to re-enter it.

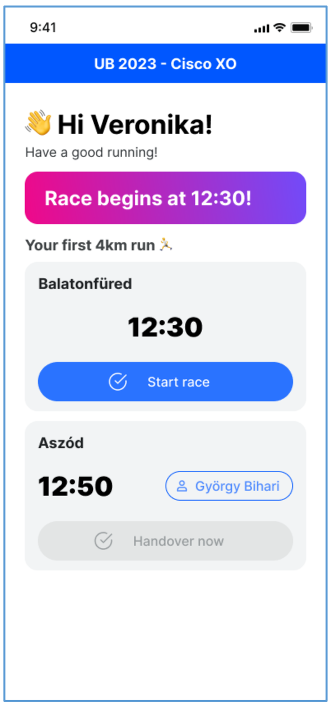
1. In the days before the race

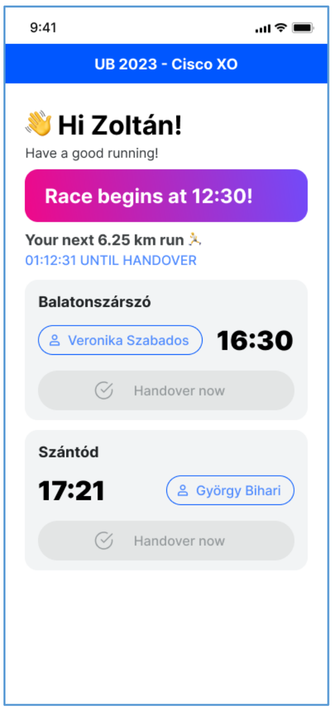


In the days leading up to the running festival, a countdown timer with the number of days remaining will be displayed. (You can get the date of the event from the “plannedStartingTime” value of the runner’s team (you may get it using /me endpoint).

2. Before the start of the race (on the day of the event)

On the day of the running festival, the display will change. It will show the planned time of the start of the team's race and the details of the runner's first run.





If the runner who is logged in as a user is running the first stage, the “Start race” button at the top of the screen will be displayed and the button will be active. For other runners the button will be “Handover now” and will not be clickable. The time remaining until the next run is also displayed for these runners. This counter should be updated continuously without refreshing the browser window. (See the description of the handovers below.)

You can get the details of the runner's next run using the **`/nextRun`** endpoint

What data does /**`nextRun`** return?

The endpoint returns data in the following format:

```
{
    "stage": {
        "id": 13,
        "startingLocation": "Badacsonytördemic",
        "arrivalLocation": "Szigliget",
        "distance": 3.4,
        "name": ""
    },
    "previousRunner": {
        "id": 3,
        "firstName": "Tibor",
        "lastName": "Nagy",
        "speed": "06:00",
        "token": "333333333",
        "isAdmin": false,
        "teamId": 1
    },
    "nextRunner": {
        "id": 1,
        "firstName": "Zoltán",
        "lastName": "Sisák",
        "speed": "06:00",
        "token": "111111111",
        "isAdmin": true,
        "teamId": 1
    },
    "canStart": false,
    "plannedStartTime": "2023-03-31T19:51:39.401055Z",
    "plannedFinishTime": "2023-03-31T20:12:03.404243Z"
}
```

As you can see, this includes the stage of the runner's next run, the **`previousRunner`** includes the details of the runner from whom he is taking over the baton, and the **`nextRunner`** shows the runner he is switching to at the end of his run.

`canStart` becomes true when the previous runner has already started his/her run.

The planned start time (**`plannedStartTime`**) and the planned finish time (**`plannedFinishTime`**) are recalculated by the backend when a handover occurs and the time of this handover is entered in the database.

If the first runner clicks on the “Start race” button, the start time will be added to the team's **`startingTime`** field by a post request to **`/handover/start`** endpoint. From this moment on, the backend calculates the **`plannedStartTime`** and **`plannedFinishTime`** values relative to this time **`(e.g. /nextRun endponint)`**

3. On the day of the race, after the start of the team

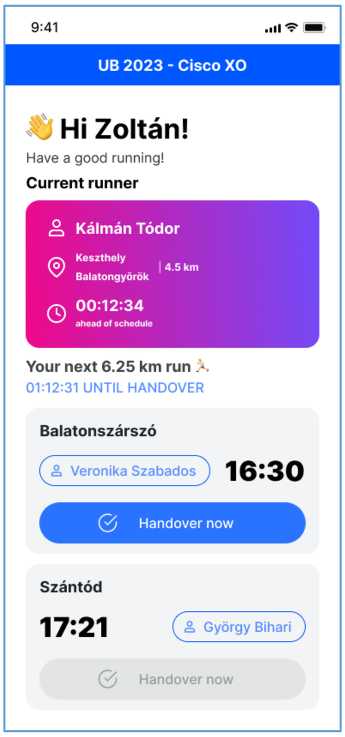
Once the team has started its running, the top third of the window shows the details of the runner currently running. This data is supplied by the backend on the /currentRunner endpoint. The time difference compared to the original schedule is returned in seconds in the "scheduleDifference" field of the endpoint. A negative value means that the team is ahead of schedule, a positive value means that the team is behind schedule.

When a team member finishes a stage, he/she needs to hand the running over to the next runner. For this, there are 2 endpoints you can use: /handover/start and /handover/finish. These endpoints always receive your stage id (the one returned by /nextRun). The start endpoint hands the running over from the previous runner to you. The finish endpoint finishes your stage, and hands it over to the next runner. This means, that both the current and the next runner can press the handover button, the result will be the same (only one of them needs to click it).

Both of these endpoints can receive a “time” property, which overwrites the date on the backend.

To keep the UI up to date, the /nextRun and /currentRunner endpoints need to be polled every 5 seconds, and after every handover action.

On the UI, there are 2 boxes that represent your stage. The first one contains the starting location of the stage, the name of the runner before you, and the planned starting time. The handover button is active, when the /nextRun endpoint returns with “canStart”: true. The second box contains information about the arrivingLocation, the expected finish time, and the name of the runner after you. The handover button is active, when the /currentRunner endpoint returns with your user id (you are the current user).


9:41

UB 2023 - Cisco XO

👋 Hi Zoltán!

Have a good running!

Current runner

Kálmán Tódor

Keszthely
Balatongyörök | 4.5 km

00:12:34
ahead of schedule

Your next 6.25 km run

01:12:31 UNTIL HANDOVER

Balatonszárszó

Veronika Szabados 16:30

Handover now

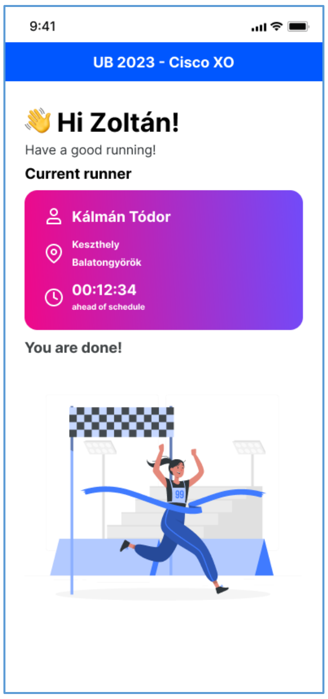
Szántód

17:21 György Bihari

Handover now

4. After the runner's last stage

When the /nextRun endpoint returns with {"finished": true}, you finished the race, and the following view should be presented:


9:41

UB 2023 - Cisco XO

👋 Hi Zoltán!

Have a good running!

Current runner

Kálmán Tódor

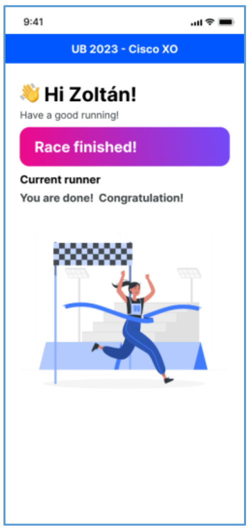
Keszthely
Balatongyörök

00:12:34
ahead of schedule

You are done!

5. After the team's run is complete

When the /currentRunner endpoint returns with {“finished”: true}, the race is over for everyone, and the following view should be presented:



👋 Hi Zoltán!

Have a good running!

Race finished!

Current runner

You are done! Congratulation!

Additional information

- You can use any resource in the assets folder.
- Clean code and user interface accessibility are also important considerations.
  - Please use semantic elements in HTML files wherever possible.
  - Provide the necessary amount of comments in your HTML, CSS and JS files.
  - Check accessibility using the Axe browser extension.
- The application must work with at least the latest versions of Google Chrome and Firefox.

Appendix A

Endpoints for Part 2 (Admin module)

Login

- **POST** /login

  Logs a runner in. Validates the runner code (token), and returns the associated runner

  Request body:

  ```
  {
    "token": "111111111"
  }
  ```

  Responses:

  200 Successful operation

  ```
  {
    "status": "success",
    "user": {
      "id": 10,
      "firstName": "Tódor",
      "lastName": "Kálmán",
      "token": "047896152",
      "isAdmin": false,
      "speed": "05:31",
      "teamId": 12
    }
  }
  ```

  401 Login failed

  ```
  {
    "status": "error",
    "message": "Login failed"
  }
  ```

Stages

- **GET** /stages

  Get all stages. Returns a list of all available stages

  Responses:

  200 Successful operation

  ```
  [
    {
      "id": 10,
      "distance": 3.5,
      "startingLocation": "Balatonalakli",
      "arrivalLocation": "Fövenyes",
      "name": "NN"
    }
  ]
  ```

Teams

- **GET** /teams/{teamId}

Get one team. Returns a team by id.

teamId: integer, numeric ID of the team to get

Responses:

200 Successful operation

```
{
  "id": 10,
  "name": "Cisco XO",
  "contactEmail": "ub2023@http-alapitvany.hu",
  "location": "Budapest",
  "plannedStartingTime": "2023-01-01T23:00:00Z",
  "startingTime": "2023-01-01T23:00:00Z"
}
```

- **PUT** /teams/{teamId}

  Update a team. Updates a team's data (admin only)

  teamId: integer, numeric ID of the team to get

  Request body:

```
{
  "name": "Cisco XO",
  "location": "Budapest",
  "contactEmail": "test@test.hu"
}
```

  Responses:

  200 Successful operation

```
{
  "id": 10,
  "name": "Cisco XO",
  "contactEmail": "ub2023@http-alapitvany.hu",
  "location": "Budapest",
  "plannedStartingTime": "2023-01-01T23:00:00Z",
  "startingTime": "2023-01-01T23:00:00Z"
}
```

- **DELETE** /teams/{teamId}

  Remove a team. Deletes a team, and all associated entities (admin only)

  teamId: integer, numeric ID of the team to get

  Responses:

  200 Successful operation

```
{
  "success": true
}
```

Runners

- **GET** /teams/{teamId}/runners

  Returns all runners in a given team.

  teamId: integer, numeric ID of the team to get

Responses:

200 Successful operation

```
[
  {
    "id": 10,
    "firstName": "Tódor",
    "lastName": "Kálmán",
    "token": "047896152",
    "isAdmin": false,
    "speed": "05:31",
    "teamId": 12
  }
]
```

- **POST** /teams/{teamId}/runners

  Creates a runner for the team (admin only)

  teamId: integer, numeric ID of the team to get

  Request body:

```
{
  "firstName": "Tódor",
  "lastName": "Kálmán",
  "speed": "05:31"
}
```

  Responses:

  200 Successful operation

```
{
  "id": 10,
  "firstName": "Tódor",
  "lastName": "Kálmán",
  "token": "047896152",
  "isAdmin": false,
  "speed": "05:31",
  "teamId": 12
}
```

- **GET** /teams/{teamId}/runners/{runnerId}

  Returns the runner with the given id

  teamId: integer, numeric ID of the team to get

  runnerId: integer, numeric ID of the runner

  Responses:

  200 Successful operation

```
{
  "id": 10,
  "firstName": "Tódor",
  "lastName": "Kálmán",
  "token": "047896152",
  "isAdmin": false,
  "speed": "05:31",
  "teamId": 12
}
```

- **PUT** /teams/{teamId}/runners/{runnerId}

  Updates a runner's data in a team (admin only) teamId: integer, numeric ID of the team to get

runnerId: integer, numeric ID of the runner

Request body:

```
{
  "firstName": "Tódor",
  "lastName": "Kálmán",
  "speed": "05:31"
}
```

Responses:

200 Successful operation

```
{
  "id": 10,
  "firstName": "Tódor",
  "lastName": "Kálmán",
  "token": "047896152",
  "isAdmin": false,
  "speed": "05:31",
  "teamId": 12
}
```

- **DELETE** /teams/{teamId}/runners/{runnerId}

  Deletes a runner's account, and removes it form the team (admin only)

  teamId: integer, numeric ID of the team to get

  runnerId: integer, numeric ID of the runner

  Responses:

  200 Successful operation

```
{
  "success": true
}
```

You do not have to implement the endpoints the line below

Schedules

- **GET** /currentRunner

  Returns information about the runner currently running, and the stage he/she is on. Also returns the time difference relative to the planned schedule.

  Responses:

  200 Successful operation

```
{
  "runner": {
    "id": 10,
    "firstName": "Tódor",
    "lastName": "Kálmán",
    "token": "047896152",
    "isAdmin": false,
    "speed": "05:31",
    "teamId": 12
  },
  "stage": {
    "id": 10,
    "distance": 3.5,
    "startingLocation": "Balatonalakli",
    "arrivalLocation": "Fövenyes",
    "name": "NN"
  },
  "scheduleDifference": -1362
}
```

- **GET** /nextRun

  Returns information about the upcoming stage (run) of the current user. If the user is currently running, it returns the current stage (always returns the first not finished stage).

  Responses:

  200 Successful operation

```
{
  "previousRunner": {
    "id": 10,
    "firstName": "Tódor",
    "lastName": "Kálmán",
    "token": "047896152",
    "isAdmin": false,
    "speed": "05:31",
    "teamId": 12
  },
  "nextRunner": {
    "id": 10,
    "firstName": "Tódor",
    "lastName": "Kálmán",
    "token": "047896152",
    "isAdmin": false,
    "speed": "05:31",
    "teamId": 12
  },
  "stage": {
    "id": 10,
    "distance": 3.5,
    "startingLocation": "Balatonalakli",
    "arrivalLocation": "Fövenyes",
    "name": "NN"
  },
  "canStart": true,
  "plannesStartTime": "2023-03-31T11:51:39Z",
  "plannesFinishTime": "2023-03-31T12:06:52Z"
}
```

- **GET** /handover/start

  Finishes the previous stage, and starts the current one. It also starts the race, if the first stage is provided as an input. It can also receive a time attribute, which overwrites the current time.

  Request body:

```
{
  "stageId": 0
}
```

Responses:

200 Successful operation

```
{
  "success": true,
  "message": "Handover successful",
  "description": "string"
}
```

- **POST** /handover/finish

  Finishes the previous stage, and starts the current one. It also starts the race, if the first stage is provided as an input. It can also receive a time attribute, which overwrites the current time.

  Request body:

```
{
  "stageId": 0
}
```

  Responses:

  200 Successful operation

```
{
  "success": true,
  "message": "Handover successful",
  "description": "string"
}
```