Web Technologies

WorldSkills 2024 National Competition

HUNGARY

Final

Module C – Admin & Backend

Submitted by:

Skills IT

Contents

Introduction

In this module you have to create a REST API backend and an admin application for a fullstack application. The REST API backend can be built in PHP/Laravel or Node.js/Express. To build the admin interface you can use a JavaScript framework (with our fully functional REST API backend) or PHP/Laravel with SSR (Server Side Rendering) technology.

Submitting your work

For Part 1 (Backend), work in either `www/backend-laravel` or `www/backend-nodejs` as per your choice.

For Part 2 – Admin Site:

- If you're working with Laravel SSR: work in `www/admin-laravel`
- If you're working with Javascript or a Javascript framework:
  - Submit your source files in `www/admin-js-src`
  - Deploy (build) your app in www/admin-js

Access your work at the following URLs (where YY is your station number):

- http://backend.ub2023-YY.hu
- http://admin.ub2023-YY.hu

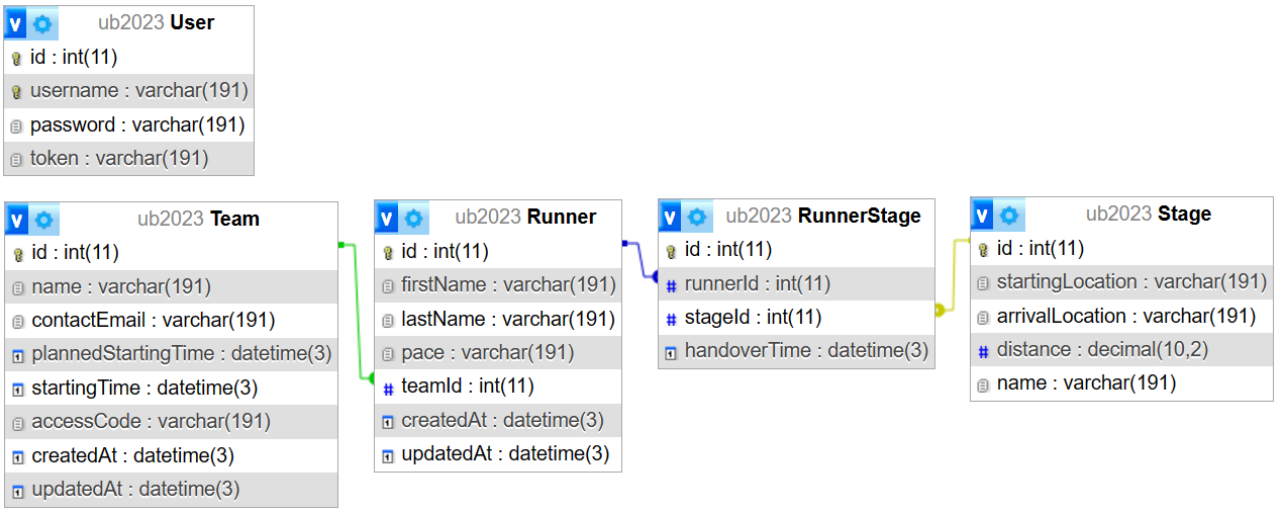
Part 1 – Backend

In this part, you need to create a REST API server module with a few endpoints. These endpoints will be used by the organiser Admin Site, as described in Part 2.

Database

The database is already available with the data already uploaded:

- Hostname: `db.ub2023.hu`
- Database: `ub2023`
- Username: `root`
- Password: `password`



An SQL dump is available in the `assets/db` folder. You can use this to restore the database to its original state.

Endpoints

The backend should serve using the following base URL: where YY is your workstation number).

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

You can also use the `backend.postman_collection.json` exported Postman collection in the `assets/postman` folder to test your endpoints.

Authentication

The application should include username-password based authentication for organiser access. The `/login` endpoint should return a randomly generated token (see Appendix A for details), and all subsequent API calls used by the Admin Site must include this token as a bearer token in the Authorization header:

```
Authorization: Bearer <token>
```

The users should be stored in the `User` table.
Create a new user in your database with the following credentials:

- Username: `admin`
- Password: `admin`

All endpoints listed in Appendix A (except `/login`) must be protected with authentication. If an incorrect token is provided, or no token is provided, the server should return with a status code of `401 Unauthorized`.

Part 2 – Organiser Admin Site

In this section, you need to create an administration module for the organisers to manage the teams and the start times of the teams.

We provide a fully working backend solution for you before this task. You can reach the endpoints of this backend with the following base URL: http://backend2.ub2023-YY.hu/api/v1, where YY is your workstation ID. See Appendix A for details of the available endpoints. You can also use the `backend2.postman_collection.json` postman collection in the `assets\postman` folder to test the endpoints and restore the database to its original state (Utility > Reset DB, /resetDb).

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.

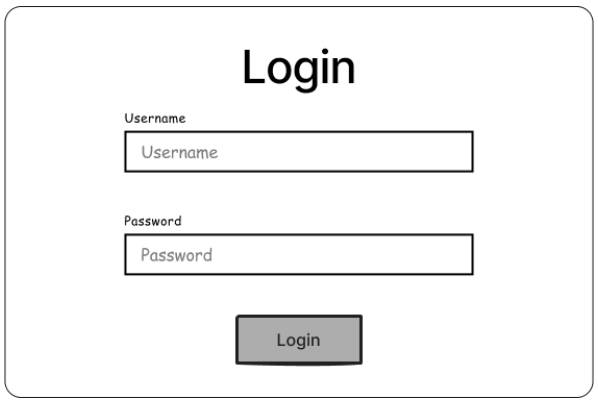
Functionalities of the Administrator Portal

- Login and logout
- Update race data
  - Update planned starting date and time
  - Update interval between planned starting times
- List and edit teams:
  - Display all registered teams
  - Update the details of a team
  - Delete a team
  - Create a team
  - Specify planned starting order and time

Administrator login

To use the admin module, the admins have to authenticate themselves by username and password. With the base URL entered into the browser, if the admin user has already logged in and has their token stored in the browser, the user will be redirected to the Teams page.

If there is no token stored, the user must enter their username and password on the Login page.



After a successful authentication, the token returned from 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 their credentials. The authenticated admin user will be redirected to the Teams page.

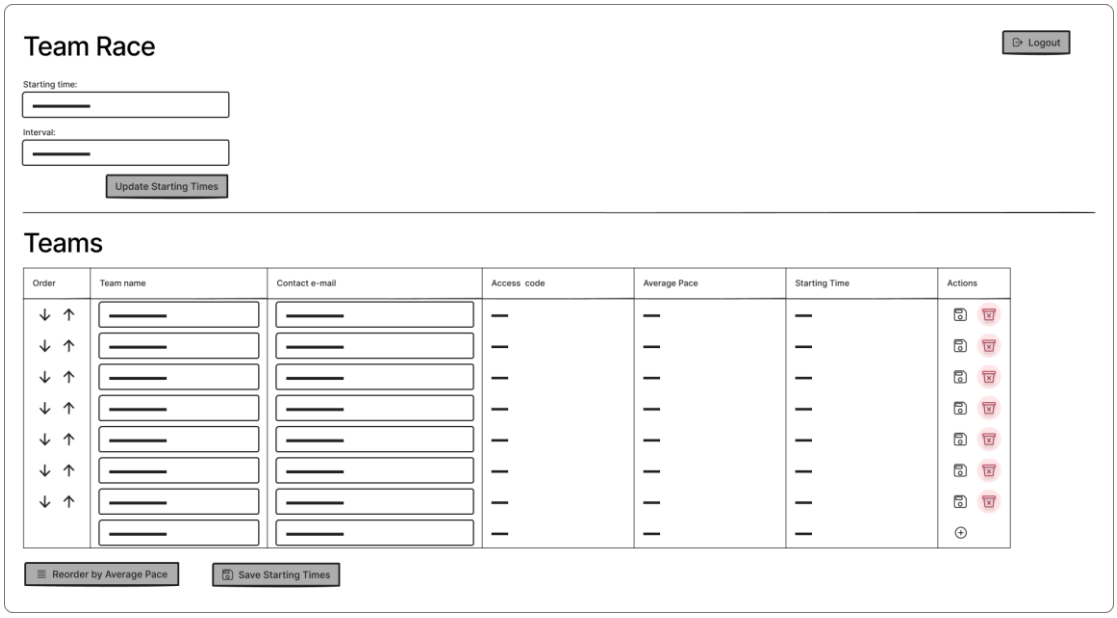
In case of unsuccessful authentication, an error message should be displayed.

One admin user is stored in the database with the following credentials:

- Username: `admin`
- Password: `admin`

All consequent requests to the backend must include the token in the Authorization header as a Bearer token. *("*`Authorization: Bearer <token>`*").* If the stored token cannot be identified by the backend (`401 Unauthorized` response from the backend), the user will be redirected back to the Login page.

Team Race page



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

The “Team Race” section at the top of the page contains two input fields:

- Starting time (the starting time of the race): this value is not explicitly stored in the database; when the page is loaded or refreshed, the earliest planned starting time among teams is loaded in this field. It should be a date-time field.
- Interval: the interval, which is the time interval between two consecutive team’s planned starting times. This value does not need to be pre-filled with a calculated value, the default value should be 5 (minutes). The user should be able to enter only integer values.

Below these, an “Update Starting Times” button is displayed. Clicking this will change the planned starting times of the teams, keeping the current order, so that the first team will start at the set starting time and the other teams will follow with the intervals set (see an example below). The field values in the Teams table have to be updated, but the changes must not be sent to the server yet. Clicking the “Save Starting Times” button on the bottom of the page saves the planned starting times to the database, but not other unsaved changes made to the teams’ data (team name and contact email).

When clicking on the up arrow at the beginning of a row, the starting position of the team in that row will be moved up one place so that its own planned starting time and the planned starting time of the team in the previous row are swapped. The order in the list will change accordingly.

A similar thing happens if you click on the down arrow. In this case, it will swap places and planned starting times with the next team after it.

These changes are also only saved to the database by clicking on the “Save Starting Times” button.

Clicking on the “Reorder by Average Pace” button defines a new order and planned starting times. The order will be determined by the average pace value. The team with the highest average pace will start first and the team with the lowest average pace will start last. The planned starting times will be recalculated according to the “Starting time” and “Interval” values in the top section.

Similarly, these changes are only saved to the database by clicking on the “Save Starting Times” button.

The “Save Starting Times” button must be disabled as long as there are no unsaved planned starting time changes. The button should be enabled once any of the planned starting times are modified in any of the ways describe above. After clicking on the button and successfully saving the planned starting times to the database, the button should be in a disabled state once again.

Example: The user sets the “Starting time” to `2023-04-24 10:00` and sets the “Interval” to `10` minutes. No changes are made yet. The user clicks the “Update Starting Times”. The first team’s “Starting Time” is set to `2023-04-24 10:00`, the second team’s to `2023-04-24 10:10`, the third one to `2023-04-24 10:20`, and so on. These values are not saved to the database yet. The

user clicks the “Save Starting Times” button at the bottom of the page, and these planned starting times are saved to the database.

The Teams section displays all registered teams with the following data:

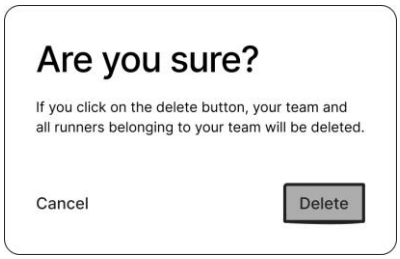
- Team name
- Contact email
- Access Code
- Average pace
- Starting time *(Planned starting time,* `plannedStartingTime` *field in the database)*

Out of these values the team’s name and the contact email are editable. Clicking on the Save icon at the end of the row saves these values to the database.
The average pace is calculated from the pace values of the team members by the backend. The Starting time is not editable directly, only in the ways described above. The access code is not editable.

The rows are sorted in an ascending order by the “Starting time” (planned starting time) value at all times.

There is also a trash icon at the end of each row. Clicking on this icon initiates the deletion of the team. After clicking, a confirmation window should appear, asking:
"Are you sure? If you click on the delete button, your team and all runners belonging to your team will be deleted.”
Clicking on “Delete” deletes the team from the database, clicking on “Cancel” closes the confirmation window.

(You can use the /resetDb endpoint, to recover the initial state after deleting a team)



When a team is deleted, the planned starting times of the other teams remains unchanged.

The last row should be an empty row where a user can create a new team. After filling in the new team’s name and contact email (both required fields), and clicking on the save button, the team is created in the database. The new team should be placed as the last in the order, with a starting time of the team before it + the interval set.

The user should only be able to create a new team if there are no unsaved starting time changes (when the “Save Starting Times” button is disabled).

Example: The current last team in order has a planned starting time of “`2023-04-24 10:40`” and the current “Interval is set to 5 (minutes). There are no unsaved planned startint time changes. After filling in the details of a new team and saving it, the new team appears in the table as the last team with a planned starting time of “`2023-04-24 10:45`”.

Appendix A – Endpoints for Part 1

Implement these endpoints in Part 1. All endpoints must be prefixed with `/api/v1`

Login

- **POST** `/login`

  Logs an organiser (admin) in. Validates the username and password; in case of success, it returns (and stores) a randomly generated token.

  - Requried header: None
  - Request body:

```
{
  "username": "admin",
  "password": "password"
}
```

  - Possible responses:
    - `200` OK (In case of successful authentication)

```
{
  "token": "b33f8c0c6a7da5e91195c372d8d9ca4b32876dc3641a764fcdc270613"
}
```

    - `401` Unauthorized (In case of invalid credentials)

```
{
  "status": "error",
  "message": "Invalid credentials"
}
```

Teams

- **GET** `/teams`

  Get a list of all teams. The `averagePace` value should be calculated by taking the simple average of the team members' paces, not taking into account the stage assignments. Members of the team who have no pace specified should not be considered. If there is no pace provided for any of the team's members, the returned `averagePace` value should be exactly "`6:00`".

  - Required header: `Authorization: Bearer <token>`
  - Possible responses:
    - `200` OK

```
[
  {
    "id": 1,
    "name": "Team 1",
```

```
    "contactEmail": "team1@example.com",
    "plannedStartingTime": "2023-04-25T10:00:00.000Z",
    "startingTime": "2023-04-25T10:02:12.000Z",
    "accessCode": "675489376",
    "averagePace": "6:01",
    "createdAt": "2023-04-20T16:00:00.000Z",
    "updatedAt": "2023-04-20T16:00:00.000Z",
  },
  ...
]
```

- 401 Unauthorized

- **POST** `/teams`

  Create a new team. All fields are required in the request body. A random `accessCode` should be generated for the new team consisting of 9 digits. The returned `averagePace` should be calculated as described above.

  - Required header: `Authorization: Bearer <token>`
  - Request body:

```
{
  "name": "Team 3",
  "contactEmail": "team3@example.com",
  "plannedStartingTime": "2023-04-25T16:00:00.000Z"
}
```

  - Possible responses:
    - 201 Created

```
{
  "id": 3,
  "name": "Team 3",
  "contactEmail": "team3@example.com",
  "plannedStartingTime": "2023-04-25T16:00:00.000Z",
  "startingTime": null,
  "accessCode": "810492116",
  "averagePace": "6:00",
  "createdAt": "2023-04-20T16:00:00.000Z",
  "updatedAt": "2023-04-20T16:00:00.000Z",
}
```

    - 400 Bad Request (in case of body validation errors)
    - 401 Unauthorized

- **PUT** `/teams/:id`

  Update a team. All values are optional in the request body. The returned `averagePace` should be calculated as described above.

  - Required header: `Authorization: Bearer <token>`
  - Request body:

```
{
  "name": "Team 1b",                                  // optional
  "contactEmail": "team1b@example.com",               // optional
  "plannedStartingTime": "2023-04-25T16:30:00.000Z"   // optional
}
```

- Possible responses:
  - 200 OK

```
{
  "id": 1,
  "name": "Team 1b",
  "contactEmail": "team1b@example.com",
  "plannedStartingTime": "2023-04-25T16:30:00.000Z",
  "startingTime": null,
  "accessCode": "810492116",
  "averagePace": "6:00",
  "createdAt": "2023-04-20T16:00:00.000Z",
  "updatedAt": "2023-04-21T10:00:00.000Z",
}
```

  - 404 Not Found

```
{
  "message": "Team not found"
}
```

  - 401 Unauthorized

- **DELETE** `/teams/:id`

  Delete a team.

  - Required header: `Authorization: Bearer <token>`
  - Possible responses:
    - 204 No Content (In case of successful deletion)
    - 404 Not Found

```
{
  "message": "Team not found"
}
```

    - 401 Unauthorized