NHL TEAM API

Henri Sjöblom

OVERVIEW

- API serves information about National Hockey League (NHL) teams
- Has four distinct enpoints: /team, /champion, /arena and /franchise
- Api has information about teams, stanley cup wins and arenas
- Franchise endpoint combines information from the other endpoints

TECHNOLOGIES

- API is based on nodeJS and express
- API has SQLite database for storing data
- Dotenv library is used to hide used PORT number to .env file
- JOI library is used for validating input data

CHALLENGES

- Designing and creating tables and data took more time than expected
- Small coding mistakes did take a lot of time to solve
- Validating input parameters was a challenge which I solved using JOI library

SOLUTIONS

- Server.js has routes and controls endpoint behaviour. It uses model.js to access database.
- Search parameters to endpoints are optional.
- JOI library is used to validate input parameters. If input parameters doesn't fullfill requirements, api returns status code 400 with error message.
- Model.js file connects to SQLite database and makes conditional queryes based on given parameters

SOLUTIONS –TEAM ENDPOINT

- Endpoint checks that search parameters are valid or undefined. Get has 3 optional search parameters. 400 returned if invalid.
- If succesfull code 200 and data returned
- If unexpected error happens code 500 and internal error message returned

```
app.get('/api/v1/team', async (req, res) => {
  const location = req.query.location
  const division = req.query.division
  const conference = req.query.conference

  const teamGetSchema = Joi.object({
    location: Joi.string().min(1),
    division: Joi.string().min(1),
    conference: Joi.string().min(1),
  })

  try {
    //validate parameters
    const { error } = teamGetSchema.validate({location, division, conference});
    if (error) {
      res.status(STATUS_BAD_REQUEST).json({ message: error.details[0].message });
      return;
    }
    const response = await findTeam(location, division, conference)

    if (response) {
      res.json(response)
    } else {
      res.status(STATUS_NOT_IMPLEMENTED).json({ message: NOT_IMPLEMENTED_MESSAGE })
    }
  } catch (error) {
      console.log(error)
      res.status(STATUS_INTERNAL_ERROR).json({ message: INTERNAL_SERVER_ERROR_MESSAGE })
  }
})
```

SOLUTIONS – ARENA ENDPOINT

- Arena endpoint has get and patch operations. Get has one optional search paratemer. The picture is from patch operation.
- Endpoint checks that given data are valid. 400 returned if invalid.
- Checks that id is in the database. If unsuccessful, 404 returned.
- If operation is succesfull, code 200 and updated entity is returned

```
app.patch('/api/v1/arena/:id', async (req, res) => {
  const id = req.params.id
  const name = req.body.name
  const teamId = req.body.teamId

  const arenaGetSchema = Joi.object({
    id: Joi.number().required().min(1),
    name: Joi.string().min(1),
    teamId: Joi.number().required().min(1)
  })

  try {
    //validate parameters
    const { error } = arenaGetSchema.validate({id, name, teamId});
    if (error) {
      res.status(STATUS_BAD_REQUEST).json({ message: error.details[0].message});
      return;
    }

    const check = await findArenaById(id)
    if (check.length === 0) {
      res.status(STATUS_NOT_FOUND).json({ message: 'Id not found' })
      return
    }
    const response = await updateArena(id, name, teamId)
    if (response) {
      const updatedEntity = await findArenaById(id)
      res.status(HTTP_STATUS_OK).json(updatedEntity)
    } else {
      res.status(STATUS_NOT_IMPLEMENTED).json({ message: NOT_IMPLEMENTED_MESSAGE })
    }
  } catch (error) {
    console.log(error)
    res.status(STATUS_INTERNAL_ERROR).json({ message: INTERNAL_SERVER_ERROR_MESSAGE })
  }
})
```

SOLUTIONS – CHAMPION ENDPOINT POST OPERATION

- Champion endpoint has get, post and delete operations. Get operation has one optional search parameter. The picture is from post operation.
- Endpoint checks that given data are valid. 400 returned if invalid.
- Checks that id is in the database. If exists, 409 returned.
- If operation is succesfull, code 201 and created entity is returned

```
app.post('/api/v1/champion', async (req, res) => {
  const id = req.body.id
  const season = req.body.season
  const teamId = req.body.team_id

  const championPostSchema = Joi.object({
    id: Joi.number().required().min(1),
    season: Joi.string().required().min(1),
    teamId: Joi.number().required().min(1)
  })

  try {

    //validate parameters
    const { error } = championPostSchema.validate({id, season, teamId});
    if (error) {
      res.status(STATUS_BAD_REQUEST).json({ message: error.details[0].message });
      return;
    }

    //check id not exists
    const check = await findChampion(id)
    if (check.length != 0) {
      res.status(POST_STATUS_CONFLICT).json({ message: 'Already exists' })
      return
    }
    const response = await createChampion(id, season, teamId)
    if (response) {
      const createdRow = await findChampion(id)
      res.status(HTTP_STATUS_CREATED).json({ createdRow })
    } else {
      res.status(STATUS_NOT_IMPLEMENTED).json({ message: NOT_IMPLEMENTED_MESSAGE })
    }
  } catch (error) {
    console.log(error)
    res.status(STATUS_INTERNAL_ERROR).json({ message: INTERNAL_SERVER_ERROR_MESSAGE })
  }
})
```

SOLUTIONS – CHAMPION ENDPOINT DELETE OPERATION

- The picture is from delete operation.
- Endpoint checks that given data are valid. 400 returned if invalid.
- Checks that id is in the database. If not, 404 returned.
- If operation is succesfull, code 200 and delete message is returned.

```
app.delete('/api/v1/champion/:id', async (req, res) => {

  const championDeleteSchema = Joi.object({
    id: Joi.number().required().min(1),
  })

  try {
    const id = parseInt(req.params.id, 10)

    //validate parameter
    const { error } = championDeleteSchema.validate({id});
    if (error) {
      res.status(STATUS_BAD_REQUEST).json({ message: error.details[0].message });
      return;
    }

    const check = await findChampion(id)
    if (check.length === 0) {
      res.status(STATUS_NOT_FOUND).json({ message: 'Id not found' })
      return
    }
    const response = await deleteChampion(id)
    if (response) {
      res.status(HTTP_STATUS_OK).json({ message: `Id ${id} deleted`})
    } else {
      res.status(STATUS_NOT_IMPLEMENTED).json({ message: NOT_IMPLEMENTED_MESSAGE })
    }
  } catch (error) {
    console.log(error)
    res.status(STATUS_INTERNAL_ERROR).json({ message: INTERNAL_SERVER_ERROR_MESSAGE })
  }
})
```

SOLUTIONS – FRANCHISE ENDPOINT

- Endpoint checks that search parameters are valid or undefined. Get has 4 optional search parameters. 400 returned if invalid.
- If succesfull code 200 and data returned
- If unexpected error happens code 500 and internal error message returned

```
app.get('/api/v1/franchise', async (req, res) => {
  const location = req.query.location
  const division = req.query.division
  const conference = req.query.conference
  const stanleyCup = req.query.stanleyCup

  const franchiseGetSchema = Joi.object({
    location: Joi.string().min(1),
    division: Joi.string().min(1),
    conference: Joi.string().min(1),
    stanleyCup: Joi.boolean()
  })

  try {
    const { error } = franchiseGetSchema.validate({location, division, conference, stanleyCup});
    if (error) {
      res.status(STATUS_BAD_REQUEST).json({ message: error.details[0].message});
      return;
    }

    const response = await findFranchise(location, division, conference, stanleyCup)
    if (response) {
      res.json(response)
    } else {
      res.status(STATUS_NOT_IMPLEMENTED).json({ message: NOT_IMPLEMENTED_MESSAGE })
    }
  } catch (error) {
    console.log(error)
    res.status(STATUS_INTERNAL_ERROR).json({ message: INTERNAL_SERVER_ERROR_MESSAGE })
  }
})
```

TEAM ENDPOINT TEST CALL

- A team endpoint test call is below
- curl --silent --include http://localhost:8080/api/v1/team?location=sunrise&conference=eastern&division=atlantic

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 103
ETag: W/"67-NwDqTCD7OiqAqx4TCJXigpDlt5M"
Date: Tue, 26 Mar 2024 07:45:55 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":13,"name":"Florida Panthers","location":"Sunrise","division":"Atlantic","conference":"Eastern"}][1]-  Done
[2]+  Done                    conference=eastern
```

ARENA ENDPOINT TEST CALLS

- A arena endpoint get test call is below
- curl --silent --include http://localhost:8080/api/v1/arena?team_id=9

```
$ curl --silent --include http://localhost:8080/api/v1/arena?team_id=9
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 84
ETag: W/"54-QqMFk4xKJBZEPkXyshY50e4LWM0"
Date: Tue, 26 Mar 2024 07:48:38 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":9,"name":"Nationwide Arena","team_id":9,"team_name":"Columbus Blue Jackets"}]
```

ARENA ENDPOINT TEST CALLS

- A arena endpoint patch test call is below
- curl -X PATCH -H "Content-Type: application/json" -d '{"name": "Colorado Center", "teamId": "8" }' --silent --include http://localhost:8080/api/v1/arena/8

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 47
ETag: W/"2f-C+7HD/YN394i9yvOpigJuiNc4sg"
Date: Tue, 26 Mar 2024 07:50:38 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":8,"name":"Colorado Center","team_id":8}]
```

CHAMPION ENDPOINT TEST CALLS

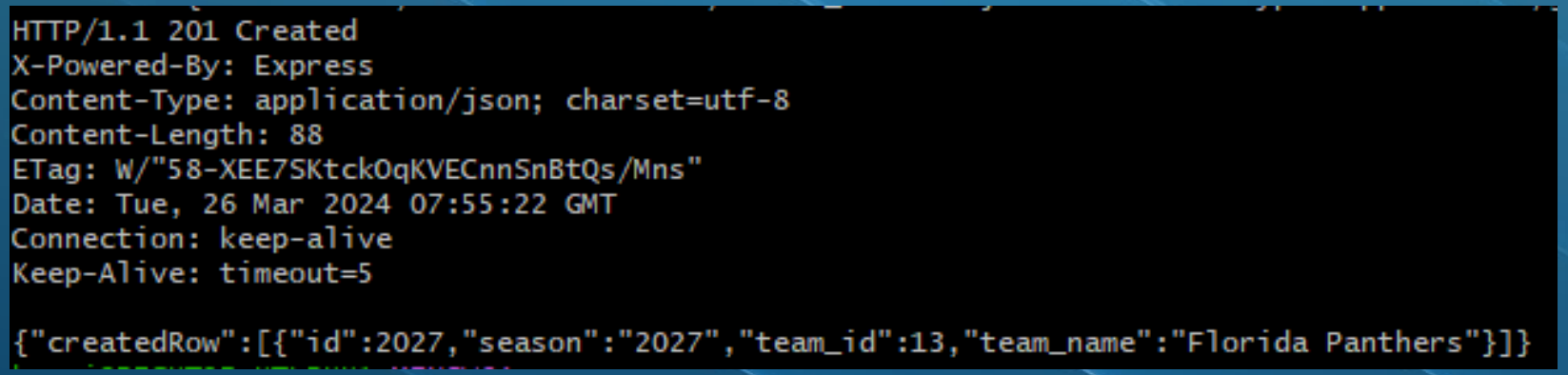
- A champion endpoint get test call is below
- curl --silent --include http://localhost:8080/api/v1/champion?season=2006&team_id=6

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 75
ETag: W/"4b-KbFen34KOd+dYBa7hNZjNjZzW8E"
Date: Tue, 26 Mar 2024 07:53:45 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":2006,"season":"2006","team_id":6,"team_name":"Carolina Hurricanes"}][1]+  Done
```

CHAMPION ENDPOINT TEST CALLS

- A champion endpoint post test call is below
- curl -d '{"id":"2027", "season": "2027", "team_id":"13"}' -H 'Content-Type: application/json' --silent --include http://localhost:8080/api/v1/champion



CHAMPION ENDPOINT TEST CALLS

- A champion endpoint delete test call is below
- curl -X "DELETE" --silent --include http://localhost:8080/api/v1/champion/2024

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 29
ETag: W/"1d-remCt6GD6uN1LV1reMR2oSHYa6E"
Date: Tue, 26 Mar 2024 07:56:20 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"message":"Id 2024 deleted"}
```

FRANCHISE ENDPOINT TEST CALL

- A franchise endpoint test call is below
- curl --silent --include http://localhost:8080/api/v1/franchise?location=chicago&conference=western&division=central&stanleyCup=true

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 154
ETag: W/"9a-P5DbbDpqXpRIy1D2XTmzK4/KGnY"
Date: Tue, 26 Mar 2024 07:58:12 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":7,"team_name":"Chicago Blackhawks","location":"Chicago","division":"Central","conference":"Western","arena_name":"United Center","stanley_cups":3}][3] 1691
[1]-  Done                    curl --silent --include http://localhost:8080/api/v1/franchise?location=chicago
[2]   Done                    conference=western
[3]+  Done                    division=central
```

TIMETABLE

- Project was started over month before the deadline.
- Mostly done during the last three days.

SUMMARY

- The NHL Team API serves information about National Hockey League (NHL) teams.
- API provides 4 endpoits
  - /team
  - /champion
  - /arena
  - franchise
- Implements CRUD operations: create, read, update and delete
- Endpoints get operations allows up to 4 search criterias.